The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015

EDITORIAL BOARD

Published twice a year by Charles University in Prague

Editorial office and subscription inquiries:
ÚFAL MFF UK, Malostranské náměstí 25, 118 00, Prague 1, Czech Republic
E-mail: pbml@ufal.mff.cuni.cz

ISSN 0032-6585

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015

CONTENTS

Articles

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015 5–20

Domain Adaptation for Machine Translation with Instance Selection

Ergun Biçici

ADAPT CNGL Centre for Global Intelligent Content
School of Computing
Dublin City University

Abstract

Domain adaptation for machine translation (MT) can be achieved by selecting training instances close to the test set from a larger set of instances. We consider 7 different domain adaptation strategies and answer 7 research questions, which give us a recipe for domain adaptation in MT. We perform English to German statistical MT (SMT) experiments in a setting where test and training sentences can come from different corpora and one of our goals is to learn the parameters of the sampling process. Domain adaptation with training instance selection can obtain 22% increase in target 2-gram recall and can gain up to 3.55 BLEU points compared with random selection. Domain adaptation with feature decay algorithm (FDA) not only achieves the highest target 2-gram recall and BLEU performance but also perfectly learns the test sample distribution parameter with correlation 0.99. Moses SMT systems built with FDA selected 10K training sentences is able to obtain F_1 results as good as the baselines that use up to 2M sentences. Moses SMT systems built with FDA selected 50K training sentences is able to obtain 1 F_1 point better results than the baselines.

1. Introduction

Machine translation (MT) performance is affected by tokens unseen in the training set, which may be due to specific use of vocabulary or grammatical structures observed in the test domain of interest. In this paper, we develop a recipe for domain adaptation for MT by comparing different strategies for the selection of training instances close to the test set from larger sets of in-domain (ID) and out-of-domain (OOD) training data. Each corpus has some characteristic distribution of vocabulary

 Corresponding author: ergun.bicici@computing.dcu.ie
Cite as: Ergun Biçici. Domain Adaptation for Machine Translation with Instance Selection. The Prague Bulletin of Mathematical Linguistics No. 103, 2015, pp. 5–20. doi: 10.1515/pralin-2015-0001.

and grammar use specific to its domain, reflected in the training instances selected for a given test corpus or for each test sentence per se. Our goal is to find the best mixture of the selected training instances in a setting where the training and test corpora can come from several different parallel corpora. We can view the test sentences as the result of a mixed selection from different domain corpora since n-grams of a sentence may come from different domains. Each test sentence defines a domain of interest that training sentences can be selected for. Therefore, the boundary between ID and OOD classes is blurred at the sentence level and in-domain or out-of-domainness is decided by a similarity function measuring the closeness of test sentences to training sentences from each domain. Each test sentence has a degree of closeness to the training domains and sampling accordingly can be a good idea. A sampling parameter for a test set specifies how much of it is selected from which domain.

Domain adaptation can be achieved by model weighting, which works with separate training and language models to obtain mixture translation models by linear combination of translation and language model probabilities with weights based on LM probabilities over training corpora split according to their genre (Foster and Kuhn, 2007). Adaptation can also be achieved by weighing the counts in the maximum likelihood estimation of phrase translation probabilities (Sennrich, 2012). Our approach is related to the instance weighting model (Foster et al., 2010). However, the instance selection models we use (Section 2) are based on scores over features consisting of n-grams in contrast to using phrases and relying on the extraction of phrase tables used during training of SMT models.

Biçici and Yuret (2011a) develop feature decay algorithm (FDA) and *dice* instance selection models, which can improve the SMT performance when compared with the performance of the SMT system using all of the training data. The results obtained demonstrate that SMT systems can improve their performance by transductive training set selection. Biçici and Yuret (2011a) focused on training instance selection for a single domain. By contrast, we demonstrate the effectiveness of instance selection for domain adaptation in a setting where test and training sets are selected from multiple separate domains generic enough to be extended to more than two domains. Previous results show that for translation at the sentence level, using only about 5000 training instances can be enough to achieve a performance close to the SMT system trained on millions of sentences (Biçici and Yuret, 2011a; Biçici, 2011).

Statistical MT (SMT) models can make use of various domain-specific training corpora to improve their performance. Adapting to a domain where parallel training resources are scarce can pose a problem for SMT performance. We provide a solution to domain adaptation with training instance selection where we retrieve relevant instances for the test set from a larger set of training instances. Our approach is transductive since we try to find training instances close to the test set and build an SMT model using the selected training set. We focus on how to pick training instances when the test set is a mixture of sentences from two different domains sampled according to a specific sampling parameter. Our goal is to closely mimic the sampling

process of the test set by creating a training set from a mixture of the two domains. We compare different MT training data selection strategies, the results of which reveal how to adapt to a new test set domain. We assume that we have two domains from which we can select training data from: domain A (D_A) and domain B (D_B). Test corpus sentences are sampled from either D_A or D_B. A sampling parameter α, $0 \leq \alpha \leq 1$, is randomly assigned to each test set where $(100 \times \alpha)\%$ of the data is selected from D_A and the rest is selected from D_B. We explore the following training data adaptation strategies:

$\mathbf{R}$	Randomly sampling from $D_A \cup D_B$.
$\mathbf{R}_\alpha$	Randomly sampling from D_A or D_B according to α.
$\mathbf{S}_{0.5}$	Selecting equally from each domain.
$\mathbf{S}_\alpha$	Selecting according to α.
$\mathbf{S}_O$	Selecting from the known, oracle, test sentence domain.
$\mathbf{S}_\cup$	Selecting from $D_A \cup D_B$.
$\mathbf{S}_{\cup\rightleftarrows}$	Selecting from $D_A \cup D_B$ using common cover link (CCL) (Section 2.4).

R_α and S_α assume that α is known beforehand, making R_α a competitive baseline. S_O assumes perfect knowledge of the domain. We can also use a classifier to predict each test source sentence's domain and select from that domain. We use this perfect classification information in the oracle setting. We select either by FDA or *dice* (Section 2) for each test sentence, which also allows us to compare their performance under different domain adaptation strategies. Each training set is the union of the training sentences selected for each test sentence. One of our goals is to understand whether the sampling parameter α, reflected in the training data selections and learn α since we can use α to adapt to a target domain.

Mandal et al. (2008) use the language model (LM) perplexity and inter-SMT-system disagreement to select training data. Moore and Lewis (2010) select training data for LMs using the difference of the cross-entropy of ID and OOD training data: $H_{ID}(s) - H_{OOD}(s)$. OOD LM training data is randomly sampled to make its size close to the ID LM training data and the vocabulary used is set to the ID vocabulary items that are observed at least twice. Axelrod et al. (2011) use bilingual cross-entropy difference:

$$\phi_{aml}(s,t) = H^S_{ID}(s) - H^S_{OOD}(s) + H^T_{ID}(t) - H^T_{OOD}(t), \quad (1)$$

where S stands for the source language, T stands for the target language, and (s,t) is a training sentence pair being scored. Lower $\phi_{aml}(s,t)$ scores correspond to better training instances. Mansour et al. (2011) use IBM Model 1 (Brown et al., 1993) and LM perplexity to filter training data and the LM corpus. We also select according to Equation (1): S_{aml}.

We answer 7 main research questions addressing how much impact does sampling parameter α have on the domain adaptation performance, whether knowing

the domain from where each test sentence is selected from helps the performance, how much instance selection improves the performance, whether we can learn α by instance selection, and what can be the best recipe for domain adaptation in machine translation:

Q1 How much does knowing α improve the random sampling performance? (R vs. R_α)
Q2 Would the performance improve if we select from the exact domain where each test instance is sampled from? (S_O vs. $S_{0.5}$ or S_α)
Q3 How much do we gain by training instance selection? (R vs. $S_{0.5}$ and R_α vs. S_α)
Q4 How much does knowing α improve the selection performance? ($S_{0.5}$ vs. S_α)
Q5 What happens if only use instance selection methods? ($S_\cup$ and $S_{\cup_\rightleftarrows}$)
Q6 Does the selection α resemble the test set α? (Correlation of α vs. $\alpha_{S_\cup}$ and $\alpha_{S_{\cup_\rightleftarrows}}$)
Q7 How should we adaptively select SMT training data for a given test domain?

We use state-of-the-art instance selection models to learn a recipe for domain adaptation. We validate our the domain adaptation approach for not only a single SMT experiment but for 1400 different SMT systems and answer 7 important research questions while comparing 7 domain adaptation strategies. Our results demonstrate that using training instance selection over all of the instances available can increase target 2-gram recall, the percentage of test target 2-grams found in the training set, by 22% and BLEU (Papineni et al., 2002) by 3.55 points. Our results may generalize to other domain adaptation tasks in natural language processing as well such as parsing.

2. Instance Selection Algorithms

We use two training instance selection models for domain adaptation: feature decay algorithms and instance selection for alignment (*dice*), where both try to increase the recall of test target features in the training set. We use a scaling parameter for selecting shorter instances having similar source and target lengths. High coverage of target features in the training sets is important for achieving high BLEU performance (Biçici, 2011).

2.1. Feature Decay Algorithm (FDA)

Feature decay algorithms (Biçici and Yuret, 2011a, 2015) increase the diversity of the training set by decaying the weights of n-gram features that have already been included and try to maximize the coverage of source side features of the test set. FDA decays the initial feature weights as instances containing them are included in the se-

lected training data where the order by which sentences are selected is determined by a sentence score which is calculated by weighted sum of feature weights. Algorithm 1 presents the FDA algorithm.

Algorithm 1: The Feature Decay Algorithm

Input: Parallel training sentences $\mathcal{U}$, test set features $\mathcal{F}$, and desired number of training instances N.
Data: A priority queue $\mathcal{Q}$, sentence scores `score`, feature values `fval`.
Output: Subset of the parallel sentences to be used as the training data $\mathcal{L} \subseteq \mathcal{U}$.

foreach $f \in \mathcal{F}$ **do**
  $\texttt{fval}(f) \leftarrow \texttt{init}(f, \mathcal{U})$
foreach $S \in \mathcal{U}$ **do**
  $\texttt{score}(S) \leftarrow \frac{1}{z} \sum_{f \in \texttt{features}(S)} \texttt{fval}(f)$
  $\texttt{enqueue}(\mathcal{Q}, S, \texttt{score}(S))$
while $|\mathcal{L}| < N$ **do**
  $S \leftarrow \texttt{dequeue}(\mathcal{Q})$
  $\texttt{score}(S) \leftarrow \frac{1}{z} \sum_{f \in \texttt{features}(S)} \texttt{fval}(f)$
  if $\texttt{score}(S) \geq \texttt{topval}(\mathcal{Q})$ **then**
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{S\}$
    foreach $f \in \texttt{features}(S)$ **do**
      $\texttt{fval}(f) \leftarrow \texttt{decay}(f, \mathcal{U}, \mathcal{L})$
  else
    $\texttt{enqueue}(\mathcal{Q}, S, \texttt{score}(S))$

We summarize the initialization, decaying, and scoring used in the FDA version.

Initialization:

$$\texttt{init}(f) = \log(|\mathcal{U}|/C_{\mathcal{U}}(f))$$

Decay:

$$\texttt{decay}(f) = \texttt{init}(f)(1 + C_{\mathcal{L}}(f))^{-1}$$

Sentence score:

$$\texttt{score}(S) = \frac{1}{z} \sum_{f \in F(S)} \textit{fvalue}(f)$$

The input to the algorithm is parallel training sentences, $\mathcal{U}$, the number of desired training instances, and the source-language features of the test set. The feature decay function, `decay`, is the most important part of the algorithm where feature weights are multiplied by $1/(1 + C_{\mathcal{L}}(f))$, where $C_{\mathcal{L}}(f)$ returns the count of f in $\mathcal{L}$, the subset of the corpus to be used as the training data. *fvalue*(.) is a function returning the weight of the argument feature. $F(S)$ returns the features of sentence S. The initialization function, `init`, calculates the log of inverse document frequency (idf), where $|\mathcal{U}|$ is the sum of the number of features appearing in the training corpus and $C_{\mathcal{U}}(f)$ is the number of times feature f appear in $\mathcal{U}$.

In the FDA version used in our experiments, we use a length scaling factor that prefers balanced shorter sentences defined as: $z = |S| \max(\frac{r|S|}{|T|}, \frac{|T|}{r|S|})$, where r is the ratio of the target-sentence length to the source-sentence length observed in the training set. FDA can be used in both transductive learning scenarios where test set is used to select the training data or in active learning scenarios where training set itself is used to obtain a sorting of the training data and select.

2.2. Using FDA5

FDA can be used to model new instance selection methods for natural language processing, information retrieval, machine translation, domain adaptation, or other related tasks where diverse and relevant selection of data is needed or phenomena with decaying feature weights are observed. FDA5 is a 5 parameter version of FDA providing a class of algorithms with feature decay and capability of modeling the behavior of other instance selection models as well (Biçici and Yuret, 2015). FDA5 is developed for efficient parameterization, optimization, and implementation of FDA. FDA5 allows a shift from developing general purpose SMT systems towards task adaptive SMT solutions.

FDA5 and instructions on how to use FDA5 are available at `github.com/bicici/FDA` and the FDA5 optimizer is available at `github.com/bicici/FDAOptimization`. The main parameters to the FDA5 algorithm are presented below:

```
-n (3): maximum n-gram order for features
-t (0): number of training words output, -t0 for no limit
-i (1.0): initial feature score idf exponent
-l (1.0): initial feature score ngram length exponent
-d (0.5): final feature score decay factor
-c (0.0): final feature score decay exponent
-s (1.0): sentence score length exponent

initial feature score: fscore0 = idf^i * ngram^l
final feature score  : fscore1 = fscore0 * d^cnt * cnt^(-c)
sentence score       : sscore  = sum_fscore1 * slen^(-s)
```

2.3. Instance Selection for Alignment

Dice's coefficient (Dice, 1945) is used as a heuristic word alignment technique giving an association score for each pair of word positions (Och and Ney, 2003). *Co-occurrence* of words in the parallel training sentences is used to retrieve sentences containing co-occurring items. Dice's coefficient score is defined as: $\mathrm{dice}(x,y) = \frac{2C(x,y)}{C(x)C(y)}$, where $C(x,y)$ is the number of times x and y co-occur and $C(x)$ is the number of times x appears in the selected training set. *dice* takes a test source sentence, S', and calculates the goodness of a training sentence pair, (S,T), by the sum of the alignment scores as in Equation (2):

$$\phi_{\mathrm{dice}}(S',S,T) = \frac{1}{z} \sum_{x \in X(S')} \sum_{j=1}^{|T|} \sum_{y \in Y(x)} \mathrm{dice}(y,T_j), \tag{2}$$

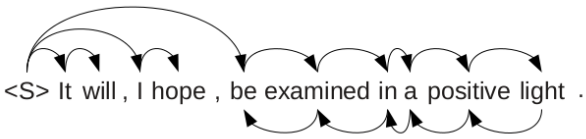


Figure 1. CCL output with arrows representing links, <S> representing the start of the sentence.

where $X(S')$ stores the features of S', $Y(x)$ lists the tokens in feature x, and

$$z = |S| \max(\frac{r|S|}{|T|}, \frac{|T|}{r|S|})(|T| \log |S| + |S| \log |T|)$$

is the scaling factor, which aims balanced shorter sentences that are not very difficult to align. $\phi_{\texttt{dice}}(S', S, T)$ favours the abundance of multiple cooccurring tokens. *dice* selects relevant training sentences for a given test sentence with a goal of improving word alignment performance (Biçici and Yuret, 2011a). SMT systems heavily rely on the word alignment of the parallel training sentences to derive a phrase table.

2.4. Features for Instance Selection

We use n-gram features when selecting training instances with up to 3-grams. We also perform unsupervised parsing using the Common Cover Link (CCL) algorithm (Seginer, 2007) and extract links from the base words to the head words. CCL allows equivalent classes with reciprocal links between words. CCL structures allow us to obtain structures representing the grammar used in the training and test sentences. Figure 1 depicts the parsing output obtained by CCL for an example sentence. Reciprocal links increase the recall and help us find relevant sentences from the training set more easily.

3. Experiments

We run experiments comparing alternative training data adaptation strategies described, which help us answer the research questions we target in Section 1. We perform translation domain adaptation experiments using the phrase-based Moses SMT system (Koehn et al., 2007). We use two parallel corpus domains: domain A (D_A) and domain B (D_B), where the training and test instances can come from. D_A uses Europarl 7 and D_B uses the News Commentary corpus. Both of these corpora are available from the WMT'12 translation challenge website (Callison-Burch et al., 2012). [1] We

[1] Parallel corpora are available from `http://www.statmt.org/wmt12/translation-task.html`

use English-German parallel corpora for our experiments and translate from English to German. D_A contains $1,920,209$ sentences and D_B contains $158,840$ sentences with average target number of words being 23.2 and 22.0, respectively.

We obtain training data by transductively selecting $10,000$ training instances to translate test sets of 100 sentences sampled according to the domain adaptation strategies. The randomly selected α values used converge to 0.5 on average. Each test set defines a new domain that we try to adapt accordingly. For each domain adaptation strategy, we perform 100 training data selection experiments. In order to obtain $10,000$ training instances for a given test set of 100 sentences, we select 100 training instances for each test sentence. This corresponds to 50 times reduction in the number of training instances selected for each sentence but doubling the training data used for translating each compared to previous work (Biçici, 2011). We focus on how to pick the training instances from separate domains when the test set is a mixture of different domain corpora. We build SMT models using Moses for each training data experiment and perform tuning over randomly selected 500 instances separately for each experiment. The LM corpus is Common Crawl from WMT'13 (Bojar et al., 2013) and it is cleaned such that sentences from D_A and D_B are excluded and fixed for all experiments. We train a 4-gram language model using SRILM (Stolcke, 2002). Out of the 1400 SMT experiments, 500 each are run with *dice* or FDA, 300 are run for random selection, and 100 are run using $50,000$ training instances selected from $D_A \cup D_B$ using FDA, corresponding to $S_{\cup 50K}$.

We obtain results that span a wide range of distributional similarities between the training and the test set. In total, we perform 1400 training data selection and SMT experiments using 18 million training, 700 thousand development, and $10,000$ test sentences. We can think of a budgeted SMT training scenario where we have a budget of \$10,000 and pay \$1 per training sentence pair used but we do not pay for searching and picking the ones we want. We are solving the following problem: given a limited budget of \$10,000, a test set of 100 sentences, and two domains to choose training instances from, how should we construct the training set for SMT? The training corpora we use is the embodiment of larger domain corpora (e.g. web crawled corpora) from which training sentences can be selected.

3.1. Training Data Comparison

Table 1 compares the training data selected with each adaptation strategy according to the average source and target recall or coverage ($scov$ and $tcov$), the number of words per sentence they contain, and the number of target 2-grams found. $scov_n$ and $tcov_n$ refer to n-gram $scov$ and $tcov$, and $scov_{\rightleftarrows}$ refer to $scov$ over CCLs. Instance selection results in shorter sentences than the randomly sampled training data but more relevant due to higher recall, the percentage of test set features found in the training set. The columns represent the number of words per sentence (wps), the number of unique 2-grams found on the target side of the training sets, and source and target

		Target Stats		scov			tcov		tcov/n-gram $\times 10^5$	
	Exp	wps	2-grams	$scov_1$	$scov_2$	$scov_{\rightleftarrows}$	$tcov_1$	$tcov_2$	1-gram	2-gram
	R	25.8	120666	.9021	.5918	.5288	.8340	.5007	3.5077	.4149
	R_α	25.4	129838	.9246	.6127	.5448	.8577	.5148	3.2065	.3980
	S_{aml}	13.7	53314	.8046	.3653	.2918	.6747	.2769	4.4230	.5989
dice	$S_{.5}$	14.8	83857	.9929	.9125	.8053	.9069	.5849	4.7626	.6978
dice	S_α	15.2	84946	.9923	.9044	.7966	.9074	.5874	4.8480	.7057
dice	S_O	9.3	47563	.9789	.8388	.7209	.8498	.4965	7.1526	1.0525
dice	$S_\cup$	13.8	76385	.9943	.9248	.8162	.9064	.5884	5.2403	.7726
dice	$S_{\cup_{\rightleftarrows}}$	14.7	78044	.9684	.8534	.8155	.8863	.5652	5.1204	.7266
FDA	$S_{0.5}$	17.4	92070	.9935	.9190	.8252	.9101	.5980	4.5641	.6500
FDA	S_α	18.3	94564	.9927	.9081	.8082	.9110	.6026	4.6085	.6491
FDA	S_O	13.9	72231	.9898	.8858	.7694	.8913	.5665	5.4169	.7937
FDA	$S_\cup$	17.7	87480	**.9947**	**.9286**	.8307	**.9127**	**.6133**	4.9965	.7037
FDA	$S_{\cup_{\rightleftarrows}}$	16.0	81570	.9696	.8630	**.8715**	.8913	.5797	5.0665	.7133
	$S_{\cup_{50K}}$	21.8	366529	.9947	.9288	.8493	.9599	.7419	1.8684	.2032
	$D_{\alpha=1}$	25.6	5645724	.9329	.8087	.7979	.9164	.7547	.7910	.0134
	$D_{\alpha=0}$	23.9	1191613	.9058	.6915	.6790	.8412	.6233	.6557	.0523

Table 1. Training data comparison for each experiment. Numbers represent averages over 100 experiments except the last two rows. Target 2-grams count the number of unique 2-grams found.

1-gram and 2-gram recall. *dice* selects relatively shorter and less diverse training sentences than FDA and obtains slightly lower recall. Both selection models improve the recall significantly. Each coverage level shows the relationship between the test domain and the training domain. We obtain baseline training data, $D_{\alpha=1}$ and $D_{\alpha=0}$, by selecting all of the training instances from D_A ($\alpha = 1$) or D_B ($\alpha = 0$), excluding the test sentences.

dice achieves similar source and target recall levels to FDA using fewer target 1-grams and 2-grams. *dice* achieves higher scores than FDA for tcov / n-gram, which calculates the target recall per the n-grams found in the training set and shows the amount of recall we achieve per n-gram used in the training set. Source recall is the result of the sentence selection process as we select by looking at the source side but target recall is unknown. The strategy $S_\cup$ lets the instance selection model find the relevant instances, which achieves the best results. We observe that additional prior knowledge about the test distribution helps (S_α); even distributing the selections equally ($S_{0.5}$) improves the performance in comparison with S_O (**Q2**, see the next paragraph). We use $\phi_{aml}(s,t)$ with strategy S_O where for each test sentence, only the domain knowledge is used. We randomly select the OOD LM as having sim-

ilar size as the ID LM corpus. The vocabulary consists of the tokens appearing at least twice in D_B. We train an open-vocabulary LM and treat tokens not appearing in the vocabulary file as <UNK>. We use ϕ_{aml} in the oracle setting as a baseline where for each test sentence, we know the domain it is coming from and accordingly we calculate ϕ_{aml} for all sentences in $D_A \cup D_B$ and sort them using Equation (1). Top $tcov_2$ is achieved with strategy $S_\cup$ using FDA. Instance selection across different domains achieve remarkable results by obtaining larger $scov$ and $tcov$ levels than either the individual domains. The ordering obtained among the strategies is given in Equation (3), which forms a recipe for domain adaptation in MT:

$$S_\cup > S_\alpha > S_{0.5} > S_{\cup_\rightleftarrows} > S_O > R_\alpha > R. \quad (3)$$

The ordering in the recipe is obtained according to statistical significance tests with paired t-testing (Wasserman, 2004) using the $tcov_2$ obtained over 100 experiments with different strategies. A $>$ represents statistically significantly better performance and $\geq$ represents better but not statistically significant improvement.

$S_{0.5}$ gives close results to S_α since we selected α randomly and on average it converges to 0.5. We are surprised to see that S_O does not give the best results and obtains the least diverse training data, which reduces its recall. S_O is restricts the domain of the training sentences selected for each test sentence to the known oracle domain whereas S_α has more freedom when selecting by benefiting from relevant instances from the other domain as well. Table 1 shows that S_O is not the best strategy. If each sentence defines a domain of interest, its features may best be utilized by a mixture selection model for domain adaptation as we observe with the $S_\cup$ strategy. $S_{\cup_\rightleftarrows}$ obtains better results than S_O but obtains lower recall than $S_\cup$, which is likely to be due to a lot of CCLs being absent from the training set. Our recipe contains the essence of domain adaptation in a single line and abstracts the results obtained with different domain adaptation strategies. FDA with $S_{\cup_{50K}}$ improves $tcov_1$ by 5 percentage points and $tcov_2$ by 13.

As α converges to 0.5 over all 100 experiments, we have identified 4 cases restricting the α selection range and looked at the closeness of the training data to the test data in Table 2 in terms of the test target 2-grams recall. $\alpha \leq 0.1$ corresponds to selecting at least 90% of test set instances from D_B and $\alpha > 0.9$ selects at least 90% of them from D_A. The $tcov_2$ differences between $\alpha \leq 0.5$ and $\alpha > 0.5$ and between $\alpha \leq 0.1$ and $\alpha > 0.9$ are larger in setting $\cup$. Setting $S_\cup$ performs best when $\alpha > 0.9$, which is expected since it contains mostly sentences from D_A. Table 2 shows that $S_\cup$ achieves the best $tcov_2$ across all α ranges for FDA and most of them for *dice*.

3.2. Translation Results

Table 3 (left) shows the translation performance using a Moses SMT system trained with each training set to translate the test sets and the baseline system results with

	Exp	$\alpha \leq 0.5$	$\alpha > 0.5$	$\alpha \leq 0.1$	$\alpha > 0.9$
	R	.4684	.5330	.4386	.5550
	R_α	.4940	.5357	.4777	.5579
	ϕ_{aml}	.3422	.2116	.3908	.1890
dice	$S_{0.5}$	.5690	.6007	.5560	.6098
dice	S_α	.5692	.6056	.5618	.6237
dice	S_O	.4736	.5193	.4575	.5410
dice	$S_\cup$	.5666	.6101	.5507	.6263
dice	$S_{\cup\rightleftarrows}$	.5421	.5883	.5234	.6048
FDA	$S_{0.5}$	.5815	.6144	.5690	.6237
FDA	S_α	.5812	.6240	**.5757**	.6472
FDA	S_O	.5431	.5898	.5304	.6078
FDA	$S_\cup$	**.5914**	**.6352**	.5742	**.6518**
FDA	$S_{\cup\rightleftarrows}$	.5565	.6029	.5394	.6173
	$S_{\cup 50K}$	.7202	.7637	.7009	.7761

Table 2. Average tcov_2 *comparison of the training data for different* α *ranges.*

$D_{\alpha=1}$ and $D_{\alpha=0}$ [2]. tcov_2 results get reflected to the BLEU performance we obtain. FDA achieves better results than *dice* and both achieve significantly better BLEU performance than random sampling baselines. The BLEU gain becomes 3.55 points versus R and 3 points versus R_α. We present the BLEU and F_1 (Biçici, 2011) performance obtained for different α ranges in Table 3 (right). FDA using the $S_\cup$ strategy achieves the top performance. Instance selection across different domains in setting $S_{\cup 50K}$ achieve remarkable results by obtaining larger F_1 score than both of the domain specific systems. The ordering obtained among the strategies is given in Equation (4):

$$S_\cup > S_\alpha \geq S_{0.5} \geq S_{\cup\rightleftarrows} \geq S_O > R_\alpha > R. \quad (4)$$

The ordering is obtained according to statistical significance tests with paired t-testing using the corpus level BLEU and F_1 (Biçici and Yuret, 2011b; Biçici, 2011) scores. $S_{\cup\rightleftarrows}$, S_α, and $S_{0.5}$ strategies achieve close performance with each other using FDA. The $S_\cup > S_\alpha \geq S_O$ result in both recipes is very important, which shows that the boundaries defining a domain are not clear cut and we are better off using a strong instance selection model over all the available training data. We plot the BLEU performance for increasing α in Figure 2 for FDA. We observe that as $\alpha \to 1$, BLEU increases due to D_A being an easier translation domain. The gap between domain adaptation with FDA and random selection results is lowest around $\alpha = 0.4$. We are also able to obtain as good as the baseline results in terms of F_1 scores using strategy $S_\cup$ and FDA. F_1 score

[2] Baseline results are not an average but the translation performance over all of the 10K test sentences.

	BLEU		F_1	
$D_{\alpha=1}$	.1866		.2458	
$D_{\alpha=0}$	.1785		.2443	
	dice	FDA	*dice*	FDA
R	.1248		.1991	
R_α	.1305		.2066	
ϕ_{aml}	.0851		.1587	
$S_{0.5}$	.1530	.1589	.2385	.2428
S_α	.1542	.1572	.2391	.2410
S_O	.1438	.1549	.2289	.2401
$S_\cup$	**.1570**	**.1603**	**.2409**	**.2442**
$S_{\cup\rightleftarrows}$	.1520	.1537	.2329	.2342
$S_{\cup 50K}$	-	**.1770**	-	**.2559**

	Exp	$\alpha \leq 0.5$	$\alpha > 0.5$	$\alpha \leq 0.1$	$\alpha > 0.9$
	R	.1157	.1339	.1031	.1361
	R_α	.1248	.1363	.1193	.1409
	ϕ_{aml}	.1035	.0885	.1071	.0850
dice	$S_{0.5}$	.1463	.1597	**.1427**	.1654
dice	S_α	.1449	.1634	.1370	.1703
dice	S_O	.1348	.1528	.1257	.1561
dice	$S_\cup$	.1488	.1652	.1385	.1724
dice	$S_{\cup\rightleftarrows}$	.1423	.1617	.1336	.1680
FDA	$S_{0.5}$	.1511	.1667	.1386	.1702
FDA	S_α	.1483	.1660	.1384	.1727
FDA	S_O	.1466	.1632	.1334	.1673
FDA	$S_\cup$	**.1520**	**.1686**	.1397	**.1772**
FDA	$S_{\cup\rightleftarrows}$	.1430	.1644	.1329	.1697
FDA	$S_{\cup 50K}$	**.1690**	**.1850**	**.1610**	**.1915**

Table 3. Average BLEU and F_1 comparison for each experiment setting and baselines (left) and average BLEU comparison for each experiment setting for different α ranges (right).

can be easily interpreted and it correlates well with human judgments (Callison-Burch et al., 2011).

3.3. Instance Selection α

We compare the test sample distribution parameter α with the α present in the selected training sets in training data adaptation strategies $S_\cup$ and $S_{\cup\rightleftarrows}$, which select from $D_A \cup D_B$. We denote the corresponding learned αs as $\alpha_{S_\cup}$ and $\alpha_{S_{\cup\rightleftarrows}}$. Test set α affects the distribution of the features in the selected training sets such that the selection α may mimic the test set α. We use α to measure a given instance selection model's effectiveness in learning the inherent α of a new test domain. Table 4 presents the mean (μ) and variance (σ) of the α values obtained. μ for α is very close to 0.5 since it is randomly selected for each of the 100 experiments. μ for the learned αs are around 0.85 with σ around 0.055. Thus, $S_\cup$ and $S_{\cup\rightleftarrows}$ tend to select about 85% of the training data from D_A. This may be expected since the size of D_A is about 12 times the size of D_B and there may be more relevant instances in D_A. But as we show in the results, the instance selection models overcome this bias and manage to select with close to perfect correlation with the actual α.

Table 4 also presents the correlation results we obtain when we compare the actual αs for all of the 100 experiments with the selected αs. The results show that we can

	μ		σ	
α	0.5059		0.278	
	dice	FDA	*dice*	FDA
$\alpha_{S_\cup}$	0.8106	0.8326	0.062	0.066
$\alpha_{S_{\cup\rightleftarrows}}$	0.8595	0.8731	0.049	0.049

r	*dice*	FDA
$r(\alpha, \alpha_{S_\cup})$	0.9864	0.9857
$r(\alpha, \alpha_{S_{\cup\rightleftarrows}})$	0.9788	0.9783

Table 4. Mean (μ) and variance (σ) of the sampling parameter α values obtained (left) and their correlation (r) (right).

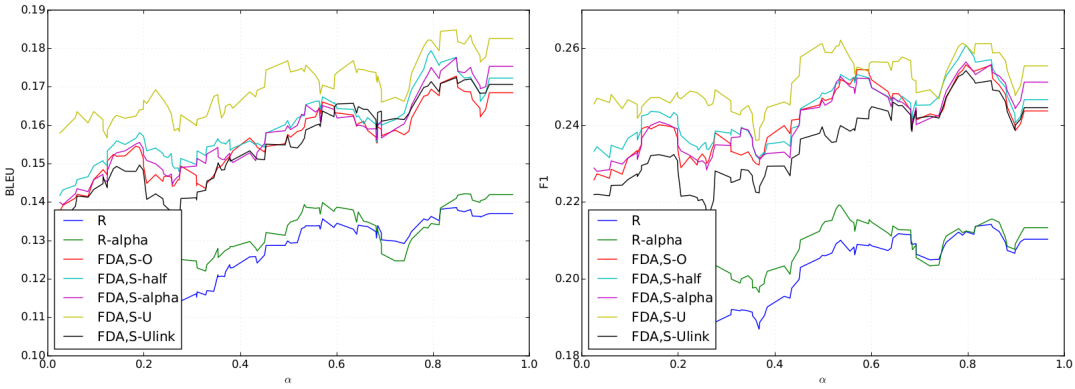


Figure 2. BLEU and F_1 for increasing sampling parameter α for FDA.

achieve close to perfect correlation with the actual αs. Thus, even though the training data adaptation strategies $S_\cup$ and $S_{\cup\rightleftarrows}$ select more from D_A and achieve larger μ for the selected αs, they perfectly correlate with the sampling parameter α. In other words, FDA and *dice* are able to mimic the sampling parameter successfully and still continue to retrieve relevant training instances at the same time.

4. Contributions

Our results answer the questions we have asked in Section 1, which we summarize below:

A1 Knowing α increases $tcov_2$ by 3% and BLEU by 0.26 points when sampling randomly.

A2 Knowing the domain of each test sentence does not improve the performance with FDA or *dice*.

A3 Instance selection can increase $tcov_2$ by 22% and BLEU by 3.55 points when compared with random sampling. Instance selection with known α increases $tcov_2$ by 20.4% and BLEU by 3.24 points.

A4 Knowing α improves BLEU by 0.3 points for *dice* and 0.1 points for FDA but does not significantly increase $tcov_2$.

A5 Instance selection without known α over all available training data using n-gram features achieves the best results with 22% increase in $tcov_2$ and BLEU gains of up to 3.55 points.

A6 Selection αs perfectly mimic the test set α with $r \sim 0.99$, which shows the effectiveness of the instance selection models.

A7 We arrive at a recipe to adapt SMT training data to a given new test domain based on the $tcov_2$ and BLEU performance: $S_{\cup} > S_{\alpha} \geq S_{0.5} \geq S_{\cup_{\rightleftarrows}} \geq S_O > R_{\alpha} > R$. Our results demonstrate that following the recipe can result in gains of up to 3.55 BLEU points and 22% increase in $tcov_2$.

Our results demonstrate that the boundaries defining a domain are not clear cut and domain selection at the corpus level or the sentence level is not as effective as sentence-level training instance selection using all of the available corpora. Each sentence defines a domain of interest and we show that its features are best utilized by a mixture selection model with strategy $S_{\cup}$ using FDA. FDA selected 10K training sentences using strategy $S_{\cup}$ is able to obtain F_1 results as good as the baseline systems using 2M sentences. FDA selected 50K training sentences is able to obtain BLEU results as good as the baseline and obtains 1 F_1 point better results. We also show that our instance selection techniques are able to perfectly learn the sampling parameter of the test set. Matching orderings in the recipes obtained according to coverage and translation performance supports that high coverage is important for achieving high BLEU performance.

We obtain remarkable results showing that instance selection across different domains achieve better $scov$ and $tcov$ than either the individual domains and better F_1 score than both of the domain specific systems in setting $S_{\cup_{50K}}$ using FDA. Our results show that sharing data across different domains is providing an advantage over competing domain specific corpora. Instance selection for domain adaptation is diminishing the competitive advantage of domain specific corpora and encouraging data sharing. We provide our SMT experiments' datasets via a link at `github.com/bicici/MTPPDAT`.

Acknowledgments

This work is supported in part by SFI as part of the ADAPT CNGL Centre for Global Intelligent Content (`www.adaptcentre.ie`, 07/CE/I1142) at Dublin City University and in part by the European Commission through the QTLaunchPad FP7 project (`www.qt21.eu`, No: 296347). We also thank the SFI/HEA Irish Centre for High-

End Computing (ICHEC, www.ichec.ie) for the provision of computational facilities and support.

Bibliography

Axelrod, Amittai, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 355–362, 2011.

Biçici, Ergun. *The Regression Model of Machine Translation*. PhD thesis, Koç University, 2011. Supervisor: Deniz Yuret.

Biçici, Ergun and Deniz Yuret. Instance selection for machine translation using feature decay algorithms. In *Proc. of the Sixth Workshop on Statistical Machine Translation*, pages 272–283, Edinburgh, Scotland, July 2011a. Association for Computational Linguistics.

Biçici, Ergun and Deniz Yuret. RegMT system for machine translation, system combination, and evaluation. In *Proc. of the Sixth Workshop on Statistical Machine Translation*, pages 323–329, Edinburgh, Scotland, July 2011b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2137.

Biçici, Ergun and Deniz Yuret. Optimizing instance selection for statistical machine translation with feature decay algorithms. *IEEE/ACM Transactions On Audio, Speech, and Language Processing (TASLP)*, 23:339–350, 2015. doi: 10.1109/TASLP.2014.2381882.

Bojar, Ondřej, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proc. of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2201.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June 1993.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, and Omer F. Zaidan. Findings of the 2011 Workshop on Statistical Machine Translation. In *Proc. of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, England, July 2011. Association for Computational Linguistics.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 workshop on statistical machine translation. In *Proc. of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June 2012. Association for Computational Linguistics.

Dice, Lee R. Measures of the amount of ecologic association between species. *Ecology*, 26(3): 297–302, 1945. ISSN 00129658. URL http://www.jstor.org/stable/1932409.

Foster, George and Roland Kuhn. Mixture-model adaptation for SMT. In *Proc. of the Second Workshop on Statistical Machine Translation*, pages 128–135, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W07/W07-0717.

Foster, George F., Cyril Goutte, and Roland Kuhn. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proc. of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 451–459, Cambridge, MA, October 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D10-1044`.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *ACL*, pages 177–180, Prague, Czech Republic, June 2007.

Mandal, A., D. Vergyri, W. Wang, J. Zheng, A. Stolcke, G. Tur, D. Hakkani-Tur, and N.F. Ayan. Efficient data selection for machine translation. In *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, pages 261 –264, Dec 2008. doi: 10.1109/SLT.2008.4777890.

Mansour, Saab, Joern Wuebker, and Hermann Ney. Combining translation and language model scoring for domain-specific data filtering. In *International Workshop on Spoken Language Translation*, pages 222–229, San Francisco, California, USA, Dec. 2011.

Moore, Robert C. and William Lewis. Intelligent selection of language model training data. In *Proc. of the Association for Computational Linguistics 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden, July 2010. URL `http://www.aclweb.org/anthology/P10-2041`.

Och, Franz Josef and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proc. of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002.

Seginer, Yoav. *Learning Syntactic Structure*. PhD thesis, Universiteit van Amsterdam, 2007.

Sennrich, Rico. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proc. of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 539–549, Avignon, France, April 2012. URL `http://www.aclweb.org/anthology/E12-1055`.

Stolcke, Andreas. Srilm - an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, pages 901–904, 2002.

Wasserman, Larry. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2004.

Address for correspondence:
Ergun Biçici
`ergun.bicici@computing.dcu.ie`
ADAPT CNGL Centre for Global Intelligent Content
School of Computing
Dublin City University
Dublin 9, Dublin, Ireland.

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015 21–41

Resources for Indonesian Sentiment Analysis

Franky, Ondřej Bojar, Kateřina Veselovská

Charles University in Prague, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics

Abstract

In this work, we present subjectivity lexicons of positive and negative expressions for Indonesian language created by automatically translating English lexicons. Other variations are created by intersecting or unioning them. We compare the lexicons in the task of predicting sentence polarity on a set of 446 manually annotated sentences and we also contrast the generic lexicons with a small lexicon extracted directly from the annotated sentences (in a cross-validation setting). We seek for further improvements by assigning weights to lexicon entries and by wrapping the prediction into a machine learning task with a small number of additional features. We observe that lexicons are able to reach high recall but suffer from low precision when predicting whether a sentence is evaluative (positive or negative) or not (neutral). Weighting the lexicons can improve either the recall or the precision but with a comparable decrease in the other measure.

1. Introduction

Sentiment analysis has gained much attention lately mostly due to its practical applications in commercial settings. The task is being widely solved not only for English, but also for many other languages, including languages with scarce evaluative data. However, we are not aware of any systematic attempts to build sentiment analysis resources for Indonesian so far, despite the increasing use of Internet by speakers of Indonesian language.

In this paper, we present our work on two types of resources for Indonesian sentiment analysis. The first one is a small collection of sentences coming from user reviews in several domains, manually annotated for sentiment. The second one is a collection of subjectivity (sentiment) lexicons built mainly by translating available English lexicons using several methods of translation. We use these resources together

 Corresponding author: bojar@ufal.mff.cuni.cz
Cite as: Franky, Ondřej Bojar, Kateřina Veselovská. Resources for Indonesian Sentiment Analysis. The Prague Bulletin of Mathematical Linguistics No. 103, 2015, pp. 21–41. doi: 10.1515/pralin-2015-0002.

and evaluate the performance of a simple lexicon-based sentiment analysis method on the annotated data.

2. Previous Work

For non-English languages, the creation of subjectivity lexicons usually takes the advantage of the availability of WordNet for the given language. This can be found in Bakliwal et al. (2012) and Pérez-Rosas et al. (2012). The work by Bakliwal et al. (2012) is for Hindi. They start with small seeds of 45 adjectives and 75 adverbs, pre-annotated with positive, negative, or objective polarity information. The seeds are expanded using Breadth First expansion by looking at the antonymy relation for opposite polarity and synonymy for the same polarity. Pérez-Rosas et al. (2012) in their work on Spanish subjectivity lexicon take the advantage of aligned synsets between WordNets of different languages to do the mapping. They get two different lexicons. A full strength lexicon is created by taking words with strong negative or positive polarity from MPQA[1] lexicon and map them to the synsets in SentiWordNet, by taking the synset with the highest negative or positive value for each word. The found synsets are mapped to Spanish WordNet. The second lexicon, a medium strength lexicon, is created by mapping the synsets in SentiWordNet with polarity scores greater than 0.5 to Spanish WordNet.

A subjectivity lexicon for Dutch adjectives is created by Smedt and Daelemans (2012) using a mixture of manual annotation and automatic expansion. The first step is to extract adjectives with high frequencies from a collection of book reviews. Seven human annotators annotate the adjectives that are previously disambiguated using CORNETTO (an extension of Dutch WordNet). Each adjective is expanded by their best nearest neighbours (handpicked by two annotators) from the list of new adjectives taken from the corpus and using cosine similarity as the measure of similarity. Each adjective is represented as a vector of top 2,500 nouns from the same corpus. Another expansion is performed by adding words from the same synset in CORNETTO, and by using the relations provided, e.g., antonymy, synonymy.

A method of creating a subjectivity lexicon for a language with scarce resources (Romanian) is introduced by Banea et al. (2008). They propose a method to create subjectivity lexicon using an online dictionary and a collection of documents. The work uses a set of subjective words called seed words to bootstrap the lexicon creation. The process runs by querying the online dictionary using these seed words. A list of extracted words returned by dictionary for each seed word is then filtered by calculating their similarity with the seed word using Latent Semantic Analysis (LSA). The LSA module is trained on Romanian corpus of half-million words. The surviving words are added to the lexicon and the process is repeated until the maximum number of iterations is reached.

[1] http://mpqa.cs.pitt.edu/

Some other approaches in subjectivity lexicon creation for non-English languages that do not utilize dictionary or thesaurus (e.g., WordNet) can be found in Maks and Vossen (2012) and Kaji and Kitsuregawa (2007). Their works can be considered as corpus-based approaches to lexicon creation. Maks and Vossen (2012) use an underlying assumption that different types of corpus posit different characteristics of subjectivity or objectivity information. They use three different corpora of Wikipedia articles, news, and comments inside the news to build Dutch subjectivity lexicon. They take words in the news and comments that are not over-used in Wikipedia articles as subjective words. The measures of over-usage of words between the corpus are calculated using log-likelihood ratio and a DIFF calculation (Gabrielatos and Marchi, 2011).

Kaji and Kitsuregawa (2007) exploit the dependencies and language structures in Japanese to extract evaluative sentences from a collection of one billion HTML documents. They use a list of cue words to detect the presence of evaluative clauses (positive or negative) in the dependency structure of the sentence. They also use layout structures such as itemization/table in HTML documents and cue words such as 'pros and cons' and 'plus and minus' to extract positive and negative evaluative sentences. From the evaluative sentences, they extract candidate phrases consisting of adjectives and adjective phrases, e.g. noun+adjective, together with their counts in positive and negative sentences. The candidates are then filtered using chi-square and PMI polarity score, and pre-defined thresholds.

3. Annotated Sentences

In this section, we describe our annotated data. The annotated sentences were taken from user reviews on *KitaReview* website[2]. We randomly selected 24 reviews and segmented them into separate sentences. The sentences were manually checked and cleaned, removing incomplete or otherwise broken ones. The final set consists of 446 sentences.

The annotation of the sentences was performed by two native speakers of Indonesian. The annotation process equipped the sentences with the following information:

- **Sentence objectivity/subjectivity**. Annotating the sentence as objective (o), i.e. factual, expressing no opinion, or subjective (s), i.e. expressing opinion.
- **Sentence polarity**. The overall polarity of the sentence, i.e. an estimate whether the sentence makes a positive (pos), negative (neg), or neutral (non) impression on the reader.
- **Evaluative Expressions**. The words in the sentences that are considered to bear positive or negative polarity are explicitly marked: "#expression@" for positive expressions, and "#expression$" for negative ones, e.g., "meskipun relatif

[2] http://www.kitareview.com

sedikit lebih #mahal$ (expensive) ... cukup #sepadan@ (worth) dengan segala kualitas masakan".

- **Two targets flag**. While in our limited annotation, we do not explicitly mark the target(s) of the valuation(s) expressed in the annotated sentences, it is not uncommon that a sentence attributes some valuation to more than one object. Sometimes the valuations can be even contradictory. In order to at least estimate how often this complication occurs, we explicitly mark sentences with two or more targets with the flag "_TWOTARG".

Table 1 shows basic statistics of the annotation. We see that around 60% of the sentences are marked as neutral. The number of negative sentences is much lower than that of the positive ones. It can also be observed that our annotators explicitly marked more positive expressions compared to the negative ones.

Table 1. Summary of Annotated Sentences

	Annotator 1	Annotator 2
Neutral Sentences	267	281
Positive Sentences	157	150
Negative Sentences	22	15
Sentences with Two Targets	30	17
# Pos Expressions (unique)	151	114
# Neg Expressions (unique)	40	33

The annotated set of sentences is stored in a plain text file, each sentence on a separate line. Tab-delimited columns contain all the information, as summarized in Table 2.

3.1. Agreement on Overall Polarity of a Sentence

We calculated the inter-rater agreement for overall polarity (sentiment) of the two annotations using the Kappa (κ) statistic. The agreement on the level of annotating the 446 sentences as neutral vs. evaluative (i.e. positive or negative but regardless which of these two classes) is 0.697.

If we restrict the set of sentences to the 140 ones where both annotators marked the sentence as evaluative, the agreement on the actual polarity (positive or negative) is higher: κ of 0.921.

4. Subjectivity Lexicons

To the best of our knowledge, there are no subjectivity lexicons (lists of positive or negative expressions) for Indonesian.

Table 2. Columns of Our Annotated Dataset

Column Name	Description
SENTENCEID	ID of the sentence
DOCID	Review document ID of the sentence
LINK	URL of the review on the original website
CATEGORY	Domain of the review
TITLE	Title of the review
REVIEWER	The author of the review
SENTENCE	The full text of the annotated sentence, including markup for evaluative expressions and the optional "TWOTARG" flag.
OBJSUBJ	Indication whether the sentence is objective (factual) or subjective (expressing an opinion)
POSNEGNON	The overall sentence polarity (sentiment): positive/negative/neutral

We created several such lexicons from English ones by (automatic) translation. The translated lexicons were then merged by intersecting or unioning them based on their source lexicon or the method of translation. The selection of the method of lexicon creation by translating was based on some limitations of the language resources available in Indonesian. In total, we produced 12 subjectivity lexicons from translation alone and 16 lexicons from merging operations.

4.1. Producing the Basic Lexicons

We used four different English subjectivity lexicons as our source lexicons as listed below:

- Bing Liu's Opinion Lexicon[3]
  It is a subjectivity lexicon created and maintained by Bing Liu (Hu and Liu, 2004). It is a list of around 6,800 entries (positive and negative combined).
- Harvard General Inquirer
  General Inquirer lexicon[4] is a list containing words and various syntactic and semantic features or categories. The positive and negative categories can be used to extract positive and negative words from the list.

[3] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[4] http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm

- MPQA (Multi-Perspective Question Answering) Subjectivity Lexicon
  MPQA[5] or OpinionFinder[6] lexicon is a subjectivity lexicon built using manual and automatic identification of evaluative words (Wilson et al., 2005). The lexicon contains words and information about their polarities, subjectivity strengths, and also their part-of-speech tags. We took the words with 'priorpolarity' tag of 'positive', 'negative', or 'both'. In the case of 'both', we put the word in both positive and negative lists.
- SentiWordNet[7]
  SentiWordNet (Baccianella et al., 2010) is a lexicon created on the basis of WordNet synsets (Miller, 1995) by assigning polarity weight (positive/negative) to the synsets. The general approach to produce the lexicons is by using a random walk model to propagate the positive and negative weight using the relationship information found in the gloss of the synsets. We took synsets that have polarity weight (positive/negative) greater than or equal to 0.5 ($>= 0.5$).

Table 3 below provides the exact numbers of positive and negative expressions in each of the English lexicons. We found duplicated entries in our English General Inquirer lexicon. The numbers without duplication are 1,637 for positive lexicon and 2,005 for negative lexicon. This duplication does not affect the resulting Indonesian lexicons, since we run de-duplication process before producing the final lexicons, see below.

We used three methods of translation to convert the extracted English lists of words into Indonesian:

- Google Translate[8]
  We simply copied and pasted all the entries from a list into the web interface of Google Translate. We translated one list at a time, e.g., positive list from SentiWordNet, with each entry separated by a newline. The translation was carried out during November 2012.
- Moses[9]
  We used Moses and our parallel corpus of 40,369 sentences (with no additional annotation) to build a small statistical machine translation system, with 38,369 sentences for training and 2,000 sentences for tuning, and using default parameters. The training data come from several domains: news, official reports, and devotional articles.

[5] http://mpqa.cs.pitt.edu/

[6] http://mpqa.cs.pitt.edu/opinionfinder/

[7] http://sentiwordnet.isti.cnr.it

[8] http://translate.google.com

[9] http://www.statmt.org/moses

- Kamus Online Bilingual Dictionary
  We used the online bilingual dictionary Kamus.net[10] to translate expressions from the English lexicons and took only the first translation as the result.

	Source Expressions		Covered by a Translation System					
			Google		Moses		Kamus	
Lexicon	Pos	Neg	Pos	Neg	Pos	Neg	Pos	Neg
Bing Liu	2,006	4,783	91%	87%	15%	6%	63%	61%
General Inquirer	1,915	2,291	99%	99%	31%	16%	82%	81%
MPQA	2,321	4,168	94%	91%	18%	8%	67%	61%
SentiWordNet	5,730	8,821	80%	73%	18%	14%	50%	41%
Intersection	470	791						
Union	7,809	12,445						

Table 3. Number of expressions extracted from English subjectivity lexicons and the extent to which they are translatable by each examined translation system.

Table 3 summarizes the coverage of each of translation systems. A term is considered non-translated if the system fails to produce any translation as well as when it copies the input verbatim to the output.

Google Translate appears to have the best coverage while our Moses (esp. due to the relatively small training data) covers the fewest items.

After the automated translation, we removed untranslated and duplicated entries. One annotator then manually checked all entries and removed translations that did not convey evaluative sense and also translations that consisted of more than one word but did not form a single multi-word expression. Table 4 shows the number of expressions for each lexicon produced.

Google translation produces the largest lexicons compared to the other translation methods. However, after manual filtering, the results retained are comparatively smaller. One of the reasons is that most of the entries are translated into phrases that are not multi-word expressions but rather e.g. clauses or clause portions.

Moses produces a small number of results since the training data were not be large enough and come from a different domain. Most of the entries from the English lexicons cannot be translated.

From the point of view of the source lexicon, one significant observation is that that SentiWordNet loses many of its entries in the filtering process. It is due to the entries from SentiWordNet that consist of a lot of specific names such as diseases, scientific names or terms, etc., that we consider as non-evaluative.

[10] `http://www.kamus.net`

Lexicon	Translation	Entries Obtained from Translation Pos	Neg	Entries after Manual Filtering Pos	Neg
Bing Liu	Google	1,147	2,589	740	1,500
General Inq		1,203	1,443	690	911
MPQA		1,429	2,426	796	1,359
SentiWordNet		3,404	4,857	873	1,205
Bing Liu	Moses	249	255	180	165
General Inq		379	245	237	130
MPQA		372	277	236	158
SentiWordNet		847	886	236	160
Bing Liu	Kamus	641	1,290	478	910
General Inq		884	1,009	536	692
MPQA		887	1,271	560	871
SentiWordNet		1,606	1,856	582	1,221

Table 4. Number of (de-duplicated) Indonesian expressions for each source lexicon and translation method before and after manual removal of wrong expressions

4.2. Merging Basic Lexicons

With the (many) baseline lexicons translated to Indonesian, we merged them by a)intersection and b) union. The basic idea of the intersection operation is to get the expressions that are agreed by different types of lexicons. The resulting lexicon should thus be smaller but with more validated expressions. On the other hand, the union operation is meant to greedily take all possible evaluative expressions. The intersection and union operations were performed on the lexicons from the same method of translation, lexicons with the same source of English lexicon, and also to all lexicons produced from translation. Table 5 shows the number of expressions for the lexicons from merging operations.

Looking at the intersection of lexicons from the same source, we can see that there is a significant drop in the number of negative expressions for SentiWordNet. We think that this is caused by the different translations provided by Google Translate and the online dictionary. The union operation, as expected, shows an increase in the number of expressions. The total number of unique entries after unioning all lexicons

[11] We exclude Moses-translated lexicons from the intersection with the source fixed and of the overall intersection because they contain too few entries.

Table 5. Positive and Negative Expressions after Intersection and Union

		Intersection[11]		Union	
Merging Lexicons of the Same…		Pos	Neg	Pos	Neg
Translation Method	Google	364	551	1256	1921
	Moses	92	78	366	246
	Online Dict	306	448	788	1565
Source	Bing Liu	330	660	932	1781
	General Inq	330	444	963	1185
	MPQA	376	619	1040	1638
	SentiWordNet	388	543	1112	1918
Merging All Lexicons Together		178	270	1557	2665

is significantly smaller than the union of their corresponding English lexicons, but still relevant considering the smaller number of expressions each lexicon has.[12]

4.3. Annotation-Based Lexicon

Since our annotation described in Section 3 include explicit markup of evaluative expressions in the sentences, we can extract a small lexicon directly from this data. In contrast to the general lexicons obtained above, this one is very much tailored to the examined domain.

5. Evaluation

We do not compare the lexicons directly to each other, but rather employ them in the practical task of predicting sentence polarity. We use a subset of our annotated sentences where the two annotators agree on the polarity as the test set. The test set consist of 380 sentences, 125 of which are labelled as positive, 13 as negative, and the remaining 242 as neutral.

5.1. Prediction Method

Given a lexicon a prediction method is needed to estimate the polarity of a given sentence. Our prediction method is very simple and identical for all the tested lexicons.

The polarity (sentiment) of a given sentence s that contains a set of positive expressions P and negative expressions N (with default weight 1.0) is predicted as:

[12]For comparison, the total number of expression of unioning all English lexicons is 7,809 for positive expression and 12,445 for negative one

$$\text{polarity}(s) = \begin{cases} \text{positive} & \sum_{p \in P} \text{weight}_{pos}(p) > \sum_{n \in N} \text{weight}_{neg}(n) \\ \text{negative} & \sum_{p \in P} \text{weight}_{pos}(p) < \sum_{n \in N} \text{weight}_{neg}(n) \\ \text{neutral} & \sum_{p \in P} \text{weight}_{pos}(p) = \sum_{n \in N} \text{weight}_{neg}(n) \end{cases} \quad (1)$$

When searching the sentences for positive and negative expressions (originating in the lexicon in question), we use the following constraints:

- **Unique Polarity.** An expression in a sentence can only be tagged with one type of polarity, either as positive or as negative expression.
- **Prioritize positive expressions.** If the lexicon lists the same expression both as positive and negative, ignore the negative one.
- **Prioritize longer expressions.** Since there was a possibility that a shorter expression is a part of a longer one, we collected the counts by first matching the longer expressions.
- **Negation.** We adapted technique presented in (Das and Chen, 2001) to handle the negation (inversion) of sentiment caused by a negation word. We used the words 'tidak', 'tak', 'tanpa', 'belum', and 'kurang' as negation words. The words that occur between the negation word and the first punctuation after the negation word were tagged with 'NOT_', e.g. 'kurang bagus gambarnya ?' (the picture is not good enough ?) to 'kurang NOT_bagus NOT_gambarnya ?'.

5.2. Performance Measures

We compare the performance of the lexicons using precision and recall of evaluative sentences:

$$\text{P-EVL} = \frac{c_{pos,pos} + c_{neg,neg} + c_{pos,neg} + c_{neg,pos}}{c_{pos,pos} + c_{neg,neg} + c_{pos,neg} + c_{neg,pos} + c_{neu,pos} + c_{neu,neg}} \quad (2)$$

$$\text{R-EVL} = \frac{c_{pos,pos} + c_{neg,neg} + c_{pos,neg} + c_{neg,pos}}{c_{pos,pos} + c_{pos,neg} + c_{pos,neu} + c_{neg,pos} + c_{neg,neg} + c_{neg,neu}} \quad (3)$$

where:

$c_{a,b}$: count of sentences with polarity a predicted as b
pos : positive
neg : negative
neu : neutral

Note that due to the small number of negative sentences in our test set, we examine the preformance only at the 'evaluative' level, i.e. we check how well the method distinguishes evaluative (positive or negative) sentences from neutral ones, disregarding the actual polarity.

5.3. Cross-Validation for Annotation-Based Lexicon

For a fair comparison, the lexicon extracted from our corpus of annotated sentences is evaluated in a 3-fold cross-validation.

The test data was randomly split into 3 sets (folds) of sentences. From each fold, we took the tagged positive and negative expressions, resulting in three lists of subjective expressions (lexicons). The prediction was then performed on each fold using the union of the lexicons taken from the other two folds. We report the average scores over the three folds.

5.4. Baseline and Oracle

In order to provide some context to our scores, we include the **Baseline** of predicting all sentences as evaluative. Marking all sentences as neutral gives P-EVL and R-EVL zero.

The **Oracle** performance is achieved if we extract the annotation-based lexicon from the complete test set and use it to predict the evaluativeness of the very same sentences.

5.5. Results

Figure 1 plots the precision and recall of all the lexicons.

The **Baseline** of marking everything as evaluative obviously has the recall of 100% and the precision is only 36%. The **Oracle**, as expected, had the highest precision (79%) and recall (90%) compared to the other type of lexicons.

The **Annotation-Based** lexicon, as cross-validated, maintains a very good precision (76%) but suffers a loss in recall, reaching only 54%.

The other observation that could be found was about the difference in performances of lexicons that were coming from different translation methods. Lexicons coming from Google translation had slightly higher precisions and recalls while Moses-translated lexicons performed worse esp. in recall.

As expected, intersecting lexicons leads to higher precisions at the expense of recall and unioning has the opposite effect. Google Translate again stands out here, bringing the highest recall when unioning across lexicon sources (u-G) and the highest precision when intersecting (i-G).

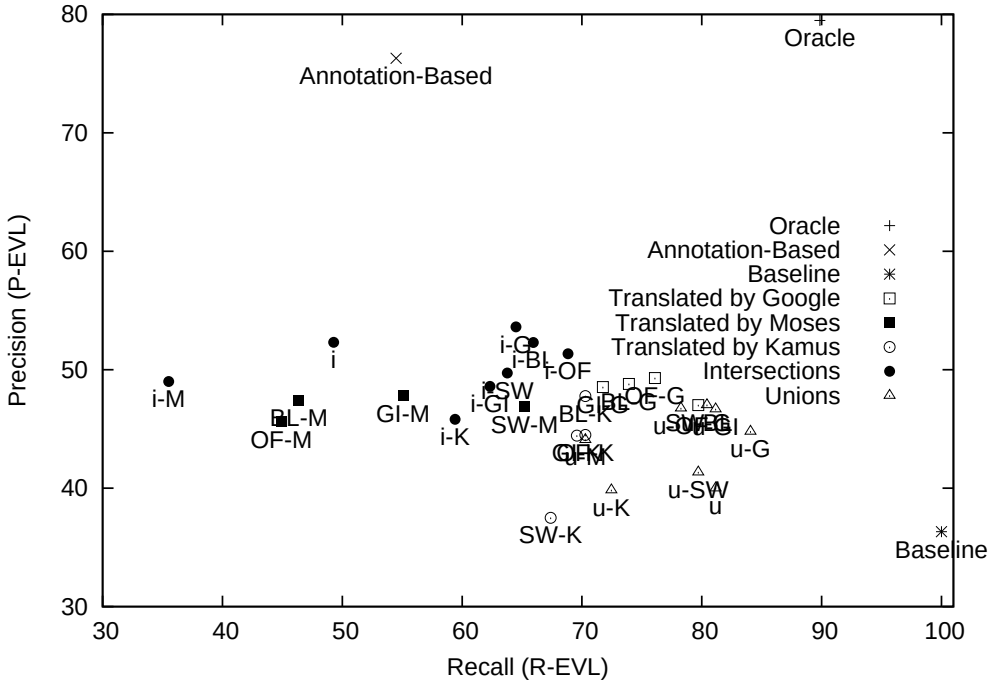


Figure 1. Precision and recall of identifying evaluative sentences using various lexicons. The ***Baseline*** *is to mark all sentences as evaluative.*

6. Weighting

The evaluations in Section 5 were done using lexicons that had expressions of weight 1.0. In realistic situation, the weight might vary, depending on how strong an expression projected the underlying positive or negative polarity.

We tried to assign this polarity strength to each expression. We used two different methods to achieve this objective. The two methods relied on the number of occurrences of the expression in a collection of 14,998 sentences coming from the same source of reviews but with no manual annotation.

The experiments in this section use the intersection of all Google-translated lexicons as the basis since this lexicon has a good balance of precision and recall.

- **Frequency Weighting**. The Frequency Weighting method assigned a weight that was the frequency or number of occurrences of the expression in the collection of the unannotated sentences. The basic premise was that the more often an expression is used in the review sentences, the higher its expressive value, assuming that the expression was used to express sentiment.

$$weight_{pos}(p) = freq_{all}(p) \quad (4)$$
$$weight_{neg}(n) = freq_{all}(n) \quad (5)$$

- **Iterative Weighting** In Iterative Weighting, an expression was given a weight of its relative frequency in the review sentences. For example, the weight a positive expression is equal to its frequency in positive sentences divided by its frequency in all of the sentences.

$$weight_{pos}(p) = freq_{pos}(p)/freq_{all}(p) \quad (6)$$
$$weight_{neg}(n) = freq_{neg}(n)/freq_{all}(n) \quad (7)$$

  Since the sentences used are unannotated sentences, we used the simple prediction method described in the previous section to first annotate the sentences. The default weight for each expression is set to 1.0. At the end of this annotation, weight of each expression is recalculated using the formula described. The prediction is repeated using these new weights, and so on until convergence.

6.1. Evaluating the Weighting Results

We showed the results of using the weighted lexicons to do prediction on test sentences in Figure 2. The accuracy and precision of lexicons with frequency and iterative weighting was lower compared to lexicon with default weight of 1.0. The significant difference was in the value of the recall. Putting weights on the expressions seemed

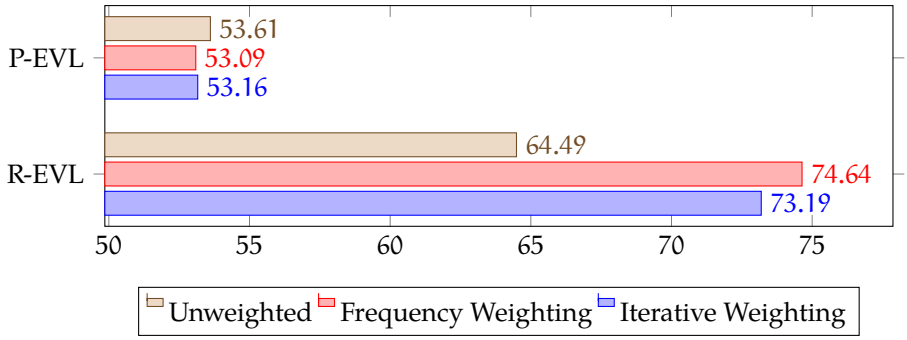


Figure 2. Impact of Frequency and Interative Weighting on evaluativeness prediction using intersection of Google-translated lexicons.

to be able to give a significant increase in the recall, with frequency weighting having higher recall than the iterative one.

The observed results confirm our expectations. Since we are evaluating only the evaluativeness of sentences and not their actual polarity, weighting has little effect on precision: sentences that contained expressions listed as positive in the lexicon will still contain them even if we reduce or increase their weight. The effect on recall can be attributed to sentences that contained an equal number of positive and negative expressions. Without weighting, the effect cancels out and the sentence is predicted as neutral. By introducing weights, we are very likely to break the balance and the sentence is predicted as evaluative one way or the other. The prediction thus marks more sentences and the growing recall confirms that these are correct sentences to mark – even humans labelled them as evaluative. If we were marking the wrong sentences, the recall would not increase and instead the precision would drop.

In Figure 3, we aim at increasing precision of the prediction. To this end, we remove expressions of low weight from the lexicon. Fewer sentences are thus going to be predicted as evaluative. Figure 3 plots the performance at the various thresholds. Only expressions with frequency higher than the threshold are included in the lexicon.

As we hoped for, excluding expressions of lower weight helps precision. However, the recall drops much faster than the precision grows.

7. Machine Learning Prediction

The simple prediction method described in Sections 5 and 6 does not consider any broader context as available in the input sentence. In general, we found several types of information that should be useful for the prediction:

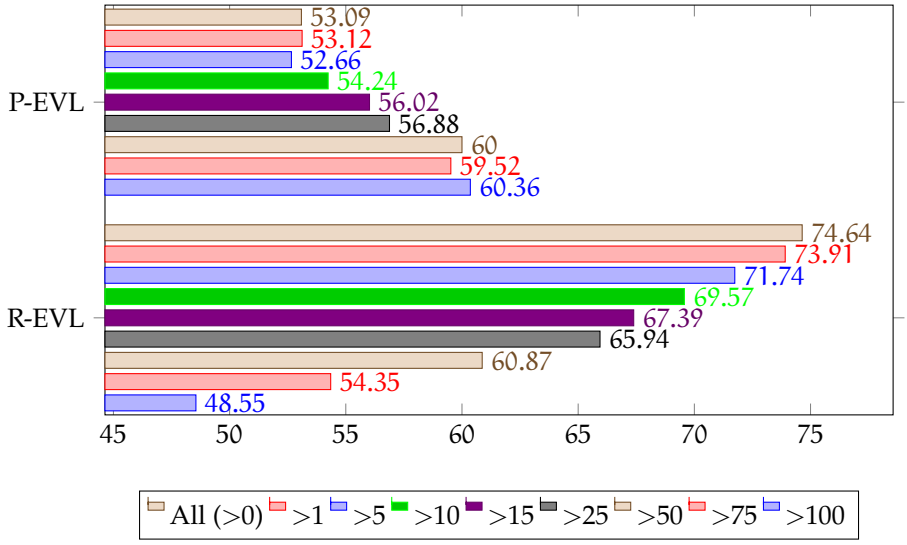


Figure 3. Thresholding using frequency weights.

- Overall Sentence Form
  A sentence with positive or negative expressions might not always be an evaluative sentence because the sentence structure or other features can prevent from such interpretation. We found several things that might cause the evaluative expressions to have no effect on the overall sentiment.
  The first case is when the sentence is in a hypothetical form, as in the example of 'sebuah keputusan yang salah akan membuat jiwa seluruh batalyon melayang percuma' (one wrong decision will cause the death of all batallions). In this sentence the word 'salah' (wrong) is identified as negative expression. However, this is only a hypothetical situation where the speaker expresses the opinion of what will happen, but not to evaluate the decision itself.
  Another structure that might affect the sentiment of the sentence is when it contrasts the positive and negative expressions as in the examples below:

  'menyuguhkan fitur yang berbeda, walau dengan model yang sama' (it comes with different features, though with the same design/model)
  'walau dengan model yang sama, menyuguhkan fitur yang berbeda' (though it has the same design/model, it comes with different features)

  In this context, the word 'berbeda' (different) is positive and 'sama' (same) is negative. Changing the parts of the sentence that are separated by a comma (one with 'though' and one without 'though') and depending on where the positive

and negative expressions are, the sentiment of the sentences can be different. The first sentence seems to be neutral and the second one seems to be more positive.
Questions are also mostly neutral, e.g., 'butuh ponsel yang murah tapi meriah?' (need a cheap and fancy phone?). The occurrence of evaluative expressions 'murah' (cheap) and 'meriah' (fancy) have no effect on the final sentiment of the sentence.

- Morphology and Multi-Word Expressions
Some other information that can be useful is related to the word itself. The first such piece of information is the part-of-speech of the word. Some evaluative expressions might have a different meaning depending on what part-of-speech they take in a sentence. For example, the word 'menarik' can have meanings of 'pull' (verb), which can be considered as having no sentiment, and 'interesting' (adjective) which has positive sentiment.
The other thing is that the evaluative expressions are sometimes used in a non-base form, e.g., 'indahnya' (how beautiful), which has the base form of 'indah' (beautiful). A simple word matching without lemmatization or stemming might not be able to capture the evaluative expression.
Words that are part of larger phrases are also tricky and might cause an inappropriate detection of evaluative expressions, e.g., 'kurang lebih' (more or less), which contains the word 'kurang' (not enough) and 'lebih' (more/better). Predictions with simple word matching that we used in previous experiments are not able to capture this phrasal information.
- Target
Information about the target of the discussion or target of the evaluation in an evaluative sentence is also important. Some sentences contain evaluative expressions that are not related to the main target of the discussion, e.g., 'selain bisa untuk berbelanja, website.com ... dengan foto-foto bayi anda yang lucu' (in addition to shopping, website.com ... with photos of your cute babies), where the target of the discussion is 'website.com' but contains a positive expression 'cute' for another target, the object 'baby'.

In this sections, we describe our experiment with machine learning techniques to include at least some of these ideas into the prediction method.

7.1. Features

Based on the previous observations, we defined a small set of 12 binary features which consisted of 10 non-lexicon related features (NonLexFeats) and 2 lexicon related features (LexFeats). The lexicon related features rely on one of the basic lexicons as used in the previous sections and they simply indicate whether at least one expression from the positive or the negative part of the lexicon was seen in the sentence.

The features are listed in Table 6.

Name	Type	Set to True if
Hypothetical	NonLex	Any of the words 'jika' (if), 'akan' (will), 'kalau' (if) appears in the sentence
Question	NonLex	'?' (question mark) appears in the sentence
Contrast 1	NonLex	Any of 'walaupun', 'meskipun', 'walau', 'meski' (though/although) is the first word of the sentence
Contrast 2	NonLex	Any of 'walaupun', 'meskipun', 'walau', 'meski' (though/although) appears anywhere except the first/last word
Negative List	NonLex	Any of the phrases 'cukup sampai disitu' (only until that point), 'kurang lebih' (more or less), 'salah satu' (one of the) appears in the sentence
Negation List	NonLex	Any of the words 'tidak' (not), 'tak' (not), 'tanpa' (without), 'belum' (not yet), 'kurang' (less), 'bukan' (is not) appears in the sentence
Adjective Word	NonLex	Any adjective (surface) words appears in the sentence
Adjective Lemma	NonLex	Any adjective lemmas appears in the sentence
Question Word	NonLex	Any question (surface) words, e.g. 'apakah' (what), 'bagaimanakah' (how), appears in the sentence
Question Lemma	NonLex	Any question lemmas, e.g. 'apa' (what), 'bagaimana' (how), appears in the sentence
PosLex	Lex	At least one of the positive expressions from the lexicon appears in the sentence
NegLex	Lex	At least one of the negative expressions from the lexicon appears in the sentence

Table 6. Features for sentiment prediction

7.2. Evaluation Setup

For the experiment, we used *scikit-learn* [13], a machine learning library for Python. We chose SVM as the machine learning method and used the default function *svm.SVC()* provided by the library. The kernel used by this function is an RBF kernel, and we just used the function with its default parameters.

The evaluation was performed using 3-fold cross validation with the same division as in Section 5. The results shown here are the average value across the three runs.

7.3. Comparing the Features

We compared the performances of using LexFeats only (lexicons only), using Non-LexFeats features only, and using all of the features (AllFeats). Figure 4 compares the average performances of these three setups. The averaging of LexFeats and AllFeats

[13] `http://scikit-learn.org`

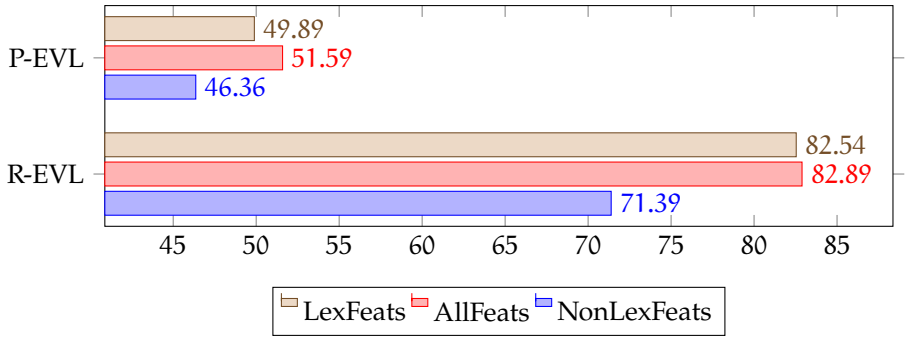


Figure 4. Average performances using various feature categories

were done across all different types of lexicons and of the three folds of test data used. The average of NonLexFeats were done only on the three folds of test data, since they were not using any lexicons in the prediction.

The results indicate that using additional features other than the lexicons improves both precision and recall, although with a rather small margin. Using only the two features of LexFeats seems to produce better results than just the NonLexFeats.

7.4. Comparison with the Simple Prediction Method

We would like to see how the machine learning prediction performance compares to the performance of the simple prediction. In order to objectively compare these two different predictions, we use the very same 3 folds for both methods and plot averaged precisions and recalls, see Figure 5.

The obvious difference that we observed was the performances of the recalls were increasing in machine learning prediction. All lexicons seemed to have high recalls, compared to the simple prediction method that had more scattered recall values. The precisions, however, showed no improvements and stayed below 60%.

8. Conclusion

We introduced two resources for Indonesian sentiment analysis: 446 annotated sentences and a collection of subjectivity lexicons constructed by manually filtering the results of automatic translation of subjectivity lexicons available for English.

The annotation of the review sentences shows the nature of the data: it mostly consists of neutral sentences. The evaluative sentences are primarily positive, so there is just a handful of negative sentences in our dataset. The inter-rater agreement (κ) for deciding whether a sentence is evaluative or not is 0.697. However, the agreement

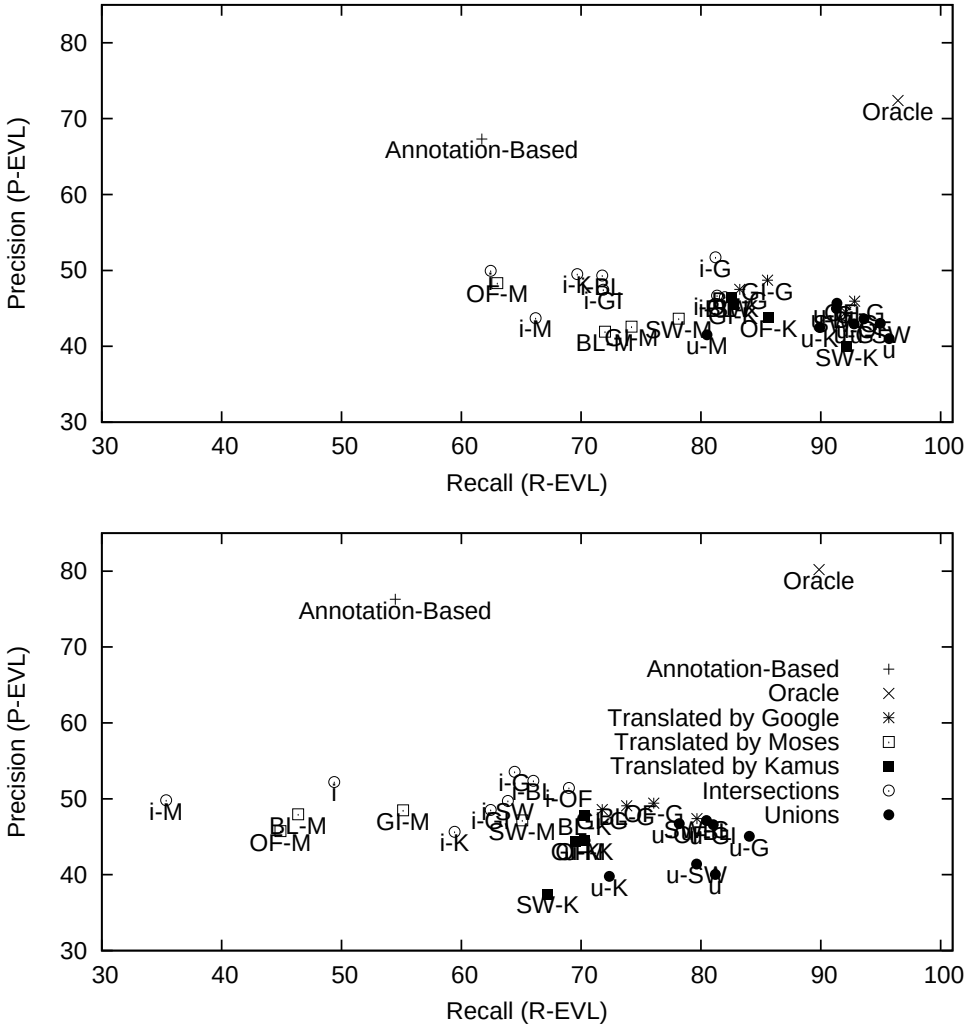


Figure 5. Precision-recall graph using our machine-learning setup (upper chart) and using the simple prediction (lower chart) in identical 3-fold cross validation.

on the actual polarity for the 140 evaluative sentences (where both annotators marked the sentence as evaluative) is surprisingly high, reaching κ of 0.921.

We produced 12 basic lexicons built by automatic translation and 16 lexicons by intersecting and unioning. The average number of expressions is 1,285 for the basic

lexicons, 747 for lexicons from intersection operations and 2,617 for lexicons from union operations.

The combination of different sources of lexicons, translation methods, and merging operations gives rise to lexicons with different numbers of entries that share some evaluative expressions but also possess their own unique expressions.

Evaluations performed on the resulting lexicons using simple prediction method show that the lexicon from intersection of Google translation of all source lexicons results in the highest precision. In terms of recall, the union of Google translations gets the highest score. The other interesting result is that a very small baseline lexicon extracted directly from (a heldout portion of) the training data achieves much higher precision than all other lexicons.

The weighting experiments that we have conducted show that the weights might help in increasing recall, although the trade-off of losing the precision exists.

We also tried to replace the basic prediction method with machine learning. This allows to incorporate other helpful information, not related to the lexicons. This helps to increase the recall for all types of the lexicons but there is no improvement in precision.

Bibliography

Baccianella, S., A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta, May. European Language Resources Association (ELRA)*, 2010.

Bakliwal, Akshat, Piyush Arora, and Vasudeva Varma. Hindi subjective lexicon: A lexical resource for hindi adjective polarity classification. In Chair), Nicoletta Calzolari (Conference, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Banea, C., R. Mihalcea, and J. Wiebe. A bootstrapping method for building subjectivity lexicons for languages with scarce resources. In *Proceedings of LREC*, 2008.

Das, Sanjiv and Mike Chen. Yahoo! for amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific Finance Association Annual Conference (APFA)*, volume 35, page 43, 2001.

Gabrielatos, Costas and Anna Marchi. Keyness: Matching metrics to definitions. *Theoretical-methodological challenges in corpus approaches to discourse studies-and some ways of addressing them*, 2011.

Hu, M. and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

Kaji, Nobuhiro and Masaru Kitsuregawa. Building lexicon for sentiment analysis from massive collection of html documents. In *Proceedings of the joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 1075–1083, 2007.

Maks, Isa and Piek Vossen. Building a fine-grained subjectivity lexicon from a web corpus. In Chair), Nicoletta Calzolari (Conference, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN 978-2-9517408-7-7.

Miller, George A. Wordnet: a lexical database for English. *Communications of the ACM*, 38(11): 39–41, 1995.

Pérez-Rosas, V., C. Banea, and R. Mihalcea. Learning sentiment lexicons in spanish. In *Proc. of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, 2012.

Smedt, T.D. and W. Daelemans. "vreselijk mooi!" (terribly beautiful): A subjectivity lexicon for Dutch adjectives. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC12)*, 2012.

Wilson, T., J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP 2005*, 2005.

Address for correspondence:
Ondřej Bojar
bojar@ufal.mff.cuni.cz
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics,
Charles University in Prague
Malostranské náměstí 25
118 00 Praha 1, Czech Republic

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015 43–64

QuEst for High Quality Machine Translation

Ergun Biçici[a], Lucia Specia[b]

[a] ADAPT CNGL Centre for Global Intelligent Content
School of Computing, Dublin City University
[b] Department of Computer Science
University of Sheffield

Abstract

In this paper we describe the use of QuEst, a framework that aims to obtain predictions on the quality of translations, to improve the performance of machine translation (MT) systems without changing their internal functioning. We apply QuEst to experiments with:

i. multiple system translation ranking, where translations produced by different MT systems are ranked according to their estimated quality, leading to gains of up to 2.72 BLEU, 3.66 BLEUs, and 2.17 F_1 points;

ii. n-best list re-ranking, where n-best list translations produced by an MT system are re-ranked based on predicted quality scores to get the best translation ranked top, which lead to improvements on sentence NIST score by 0.41 points;

iii. n-best list combination, where segments from an n-best list are combined using a lattice-based re-scoring approach that minimize word error, obtaining gains of 0.28 BLEU points; and

iv. the ITERPE strategy, which attempts to identify translation errors regardless of prediction errors (ITERPE) and build sentence-specific SMT systems (SSSS) on the ITERPE sorted instances identified as having more potential for improvement, achieving gains of up to 1.43 BLEU, 0.54 F_1, 2.9 NIST, 0.64 sentence BLEU, and 4.7 sentence NIST points in English to German over the top 100 ITERPE sorted instances.

1. Introduction

QuEst is a quality estimation framework that offers a wide range of feature extractors that can be used to describe source and translations texts in order to build

 Corresponding author: ergun.bicici@computing.dcu.ie
Cite as: Ergun Biçici, Lucia Specia. QuEst for High Quality Machine Translation. The Prague Bulletin of Mathematical Linguistics No. 103, 2015, pp. 43–64. doi: 10.1515/pralin-2015-0003.

and apply models to predict the quality of translations. It was developed within QT-LaunchPad (Preparation and Launch of a Large-Scale Action for Quality Translation Technology),[1] a project aimed at high quality machine translation through, among other things, the use of novel metrics to systematically measure and estimate translation quality.

We use QuEst to predict and improve the quality of MT systems without changing their internal functioning and evaluate with automatic evaluation methods. In what follows we describe the experimental settings (Section 2) and results of several experiments focusing on four approaches: (i) multiple system translation ranking (Section 3), (ii) n-best list re-ranking and (iii) n-best list combination (Section 4), and (iv) ITERPE to identify translations with potential for improvement and build sentence-specific SMT systems (Section 5). SMT performance improvements according to all these approaches are summarized in Table 16.

2. Experimental settings

2.1. Datasets

The multiple MT system translation ranking experiments in Section 3 use the following datasets where multiple machine translations are available for each source sentence:

DEAMT09 English to Spanish translations by four SMT systems, denoted by s_1-s_4, scored for post-editing effort (PEE) [2] in 1-4 (highest-lowest) in absolute terms (Specia et al., 2009). 3,095 sentences are used for training and 906 for testing.

DQET13-HTER English to Spanish translations scored for HTER with 2,254 sentences for training and 500 for testing (Task 1.1 dataset used in quality estimation task (QET13) at WMT13 (Bojar et al., 2013)).

DQET13-rank(de-en) German to English set of up to five alternative translations produced by different MT systems human ranked relative to each other according to their quality. 7,098 source sentences and 32,922 translations are used for training and 365 source sentences and 1,810 translations for testing (Task 1.2 dataset in QET13).

DQET13-rank(en-es) English to Spanish DQET13-rank dataset with 4,592 source sentences and 22,447 translations for training and 264 source sentences and 1,315 translations for testing.

The re-ranking and combination experiments in Section 4 use the following datasets:

DQET13-nbest English to Spanish n-best lists provided in Task 1.1 of QET13.

[1] http://www.qt21.eu

[2] as perceived by the post-editors

DFDA13-nbest English to Spanish and Spanish to English distinct 1000-best lists from Moses (Koehn et al., 2007) SMT systems developed for the WMT13 translation task using FDA5 (Biçici, 2013a; Biçici and Yuret, 2015), which is developed for efficient parameterization, optimization, and implementation of state-of-the-art instance selection model feature decay algorithms (FDA) (Biçici, 2011; Biçici and Yuret, 2015). FDA try to increase the diversity of the training set by decaying the weights of n-gram features from the test set.

The ITERPE and SSSS experiments in Section 5 use the following datasets:

DFDA14-train English to German and German to English translations of separate $3,000$ sentences randomly selected from the development sentences available at WMT14 that are unused when training the Parallel FDA5 Moses SMT systems (Biçici et al., 2014) with translations obtained using the Parallel FDA5 Moses SMT systems.

DFDA14-test English to German with $2,737$ sentences and German to English with $3,003$ sentences WMT14 translation task test set with baseline translations obtained with the Parallel FDA5 Moses SMT systems developed for the WMT14 translation task (Biçici et al., 2014).

ParFDA5 WMT14 dataset available at `https://github.com/bicici/ParFDA5WMT` provides training data for building Parallel FDA5 Moses SMT systems used. Other datasets used for the experiments, as well as the QuEst open source QE toolkit, are available for download at `http://www.quest.dcs.shef.ac.uk/`.

2.2. Evaluation metrics

We evaluate the learning performance with root mean squared error (RMSE), mean absolute error (MAE), relative absolute error (RAE), MAE relative (MAER), mean RAE relative (MRAER). RAE measures the absolute error relative to the absolute error of the mean target value, where y_i represents the actual target value for instance i, $\bar{y}$ the mean of all these instances, and $\hat{y}_i$ a prediction for y_i:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \quad \mathrm{RAE} = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{\sum_{i=1}^{n} |\bar{y} - y_i|} \tag{1}$$

We define MAER and MRAER for easier replication and comparability with relative errors for each instance:

$$\mathrm{MAER}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{\lfloor |y_i| \rfloor_\epsilon}}{n} \quad \mathrm{MRAER}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{\lfloor |\bar{y} - y_i| \rfloor_\epsilon}}{n} \tag{2}$$

MAER is the mean absolute error relative to the magnitude of the target and MRAER is the mean relative absolute error relative to the absolute error of a predictor always predicting the target mean assuming that target mean is known. MAER and MRAER are capped from below[3] with $\epsilon = \text{MAE}(\hat{\mathbf{y}}, \mathbf{y})/2$, which is the measurement error and it is estimated as half of the mean absolute error or deviation of the predictions from target mean. ϵ represents half of the score step with which a decision about a change in measurement's value can be made. ϵ is similar to half of the standard deviation, σ, of the data but over absolute differences. For discrete target scores, $\epsilon = \frac{\textit{step size}}{2}$. A method for learning decision thresholds for mimicking the human decision process when determining whether two translations are equivalent is described in (Biçici, 2013b).

Additionally, acc (accuracy) represents the percentage of source sentences for which the first-ranked translation by the ranker model agree with humans. For correlation with human judgments, we use Kendall's τ (Bojar et al., 2013). Translation performance is evaluated using BLEU (Papineni et al., 2002), F_1 (Biçici and Yuret, 2011; Biçici, 2011), 1-WER (WER for word error rate), and averaged sentence-level scores BLEUs [4] and NISTs (sentence NIST (Doddington, 2002)). F_1 has been shown to correlate with human judgments better than TER (Biçici and Yuret, 2011; Callison-Burch et al., 2011). Predicting F_1 also allowed us to achieve top results in DQET13-rank (Biçici, 2013b).

2.3. Algorithms

We use Support Vector Regression (SVR) (Smola and Schölkopf, 2004) as the learning algorithm and also use Partial Least Squares (PLS) or feature selection (FS). Feature selection is based on recursive feature elimination (RFE) (Guyon et al., 2002; Biçici, 2013b). We use `scikit-learn`[5] implementation. Some of the results may be rounded with `round(.)` function from `python`[6] and some with `numpy.round()` function from `numpy`[7], which may cause differences at the least significant digit[8].

2.4. QuEst Quality Estimation Features

QuEst offers a number of MT system- and language-independent features, of which we explore two sets:

[3] We use $\lfloor \, . \, \rfloor_\epsilon$ to cap the argument from below to ϵ.

[4] If an n-gram match is not found, the match count is set to $1/2|T'|$ where $|T'|$ is the length of the translation.

[5] `http://scikit-learn.org/`

[6] `https://www.python.org/`

[7] `http://www.scipy.org/`

[8] For instance, $\texttt{round}(0.8445, 3) = 0.845$ and $\texttt{numpy.round}(0.8445, 3) = 0.84399999999999997$.

IR		Source	Translation	Source and Translation
	Retrieval	15	15	0
Readability	LIX	1	1	0
	Word	1	1	1

Table 1. Counts of features in IR set.

- **BL**: 17 baseline features. These include sentence and average token lengths, number of punctuation symbols, LM probability, average number of translations per source word, and percentage of low or high frequency words.[9]
- **IR**: 35 information retrieval and readability features. Information retrieval features measure the closeness of the test source sentences and their translations to the parallel training data available indexed with Lucene (The Apache Software Foundation, 2014) to predict the difficulty of translating each sentence or finding their translations (Biçici et al., 2013; Biçici, 2013b). For the top five retrieved instances, retrieval scores, BLEU, and F_1 scores over the source sentence or its translation are computed quantifying the closeness of the instances we can find or their similarity to the source sentence or their translation. Readability features attempt to capture the difficulty of translating a sentence by computing the LIX readability score (Björnsson, 1968; Wikipedia, 2013) for source and target sentences, the average number of characters in source and target words, and their ratios. Table 1 shows the number of features in IR categorized according to information source.

The combined feature set, BL+IR, contains 52 features.

For experiments in Section 4, we only consider IR on the translations, since the source sentence is the same for all translation candidates. These results in 18 features, derived for each translation, using retrieval scores, BLEU, and F_1 scores over the top five instances retrieved and three LIX readability features. In that case, the combined feature set, BL+IR, contains 35 features. In those experiments, we also use Moses SMT model-based features, which are obtained from the n-best lists generated, adding 15 more features (6 for lexical reordering, 1 for distortion, 1 for language model, 1 for word penalty, 1 for phrase penalty, 4 for the translation model, and 1 for the overall translation score). This feature set is referred to as SMT. IR+SMT contains 33 features, while BL+IR+SMT contains 50 features.

3. Multiple System Translation Ranking

In multiple MT system translation ranking, we rank translations produced by different MT systems according to their estimated quality. System combination by multiple MT system translation ranking can lead to results that are better than the best

[9] http://www.quest.dcs.shef.ac.uk/quest_files/features_blackbox_baseline_17.

	DEAMT09			DQET13-HTER		
	BL	IR	BL+IR	BL	IR	BL+IR
target		F_1			F_1	
RMSE	0.1356	0.0868	**0.0847**	0.1754	0.1777	**0.1743**
MAE	0.1012	0.0535	**0.0521**	**0.1173**	0.122	0.1193
RAE	0.8514	0.4499	**0.4383**	**0.9217**	0.9589	0.9377
MAER	0.5077	0.3073	**0.3068**	**0.6771**	0.7356	0.7211
MRAER	0.8066	0.4778	**0.4612**	**0.7737**	0.8045	0.8055
target		PEE			HTER	
RMSE	0.718	0.775	**0.6772**	**0.1771**	0.1888	0.1778
MAE	0.5727	0.6291	**0.5356**	**0.1426**	0.154	0.1431
RAE	0.7045	0.7738	**0.6588**	**0.9512**	1.0268	0.9544
MAER	0.3209	0.3538	**0.2912**	1.0149	1.0865	**0.9967**
MRAER	0.7103	0.7732	**0.6758**	**0.9144**	0.9816	0.9467

Table 2. Prediction results on the DEAMT09 and DQET13-HTER datasets using a single general SVR.

MT system performance (Biçici and Yuret, 2011; Biçici, 2011). Some of the results in this Section are also presented in (Specia et al., 2013). For DEAMT09, we predict F_1 scores and PEE and for DQET13-HTER, F_1 and HTER. Table 2 presents the prediction results on DEAMT09 for translations from all four systems s_1-s_4 and on DQET13-HTER using a single general SVR model, i.e., a model combining translations from all MT systems. RAE decreases with the addition of the IR; F_1 is easier to predict than PEE when IR is included.

Table 3 presents the prediction results on the DEAMT09 datasets using separate SVR models for each translation system. In Section 3.2, we observe that building separate SVR models for each translation system achieves better performance than building a single model over all of the training set available. Table 3 shows that translation system s_4 is the easiest to predict. This is the MT system with the lowest translation performance (Table 4). IR achieve better performance when predicting F_1, but slightly worse performance when predicting PEE scores. Individual models perform better than using a general model during prediction and as we see in Section 3.1, also when ranking alternative translations for the same source sentence.

target		s_1 BL	s_1 IR	s_1 BL+IR	s_2 BL	s_2 IR	s_2 BL+IR	s_3 BL	s_3 IR	s_3 BL+IR	s_4 BL	s_4 IR	s_4 BL+IR
F_1	RMSE	.1478	.0963	.0930	.1383	.0901	.0871	.1336	.0912	.0883	.0905	.0593	**.0585**
	MAE	.1147	.0607	.0584	.1061	.0562	.0540	.1012	.0564	.0545	.0671	.0371	**.0370**
	RAE	.9161	.4850	.4663	.8875	.4704	**.4516**	.9055	.5040	.4870	.9217	.5099	.5081
	MAER	.5330	.3272	.3201	.5136	.3132	.3082	.5222	.3287	.3229	.4887	**.2989**	.3028
	MRAER	.8962	.5074	.4864	.8387	.4774	**.4595**	.8698	.5244	.5066	.9420	.5367	.5344
PEE	RMSE	.6177	.6173	**.5817**	.6764	.675	.6499	.6679	.6538	.6243	.6792	.6026	.5822
	MAE	.4490	.4712	.4428	.5423	.5456	.5215	.5280	.5238	.4976	.3448	.3216	**.3162**
	RAE	.8669	.9099	.855	.7960	.8009	.7655	.7828	.7765	.7377	.6903	.6439	**.6331**
	MAER	.1979	.2094	.1901	.2741	.2706	.2579	.2645	.2601	.2449	**.1556**	.1605	.1621
	MRAER	.5267	.5882	.5735	.7653	.7837	.7581	.7855	.7800	.7553	**.4085**	.4109	.4106

Table 3. Prediction performance of individual SVR models on the DEAMT09 dataset.

3.1. 1-best Translations

In this section the goal is to rank alternative translations based on their predicted quality to select the best MT system for each translation. For comparison, Table 4 shows the individual performance of each MT system and oracle MT system selection results based on true sentence-level scores (PEE, BLEU, and F_1). Oracle selection using PEE (human) scores obtains worse scores than s_1, the top system, which indicates that PEE does not correlate well with BLEU or F_1.

System	BLEUs	BLEU	F_1
s_1	**0.3521**	**0.3795**	**0.3723**
s_2	0.3156	0.3450	0.3361
s_3	0.2905	0.3145	0.3137
s_4	0.1600	0.1910	0.2148
oracle PEE	0.3362	0.3678	0.3574
oracle BLEU	**0.3941**	**0.4132**	0.4014
oracle F_1	0.3932	0.4130	**0.4020**

Table 4. Performance of systems in DEAMT09.

The predicted scores for each alternative translation of a given source sentence are used to rank these alternatives. For the DQET13-HTER dataset we treat relative 5-way rankings as absolute scores in $[1, 5]$ (best-worst). The absolute scores, $[1 - 4]$ for DEAMT09 and $[1-5]$ for DQET13-HTER, are referred to as PEE scores. We also predict each translation's F_1 score where y is calculated using the reference translations.

Table 5 presents the 1-best translation results for the DEAMT09 dataset obtained by ranking translations from each system based on the predictions produced by a single general SVR model or individual SVR models (Specia et al., 2010), i.e., a model built for each SMT system. In case of ties, we randomly pick among the equally scoring system outputs. As a baseline, we compute acc-best (accuracy of best-system), which is the percentage of source sentences for which the best system on average (s_1) actually provides the best translation. We achieve gains up to 2.72 BLEU, 3.66 BLEUs, and

		General			Individual		
Target	Evaluation	BL	IR	BL+IR	BL	IR	BL+IR
F1	BLEU	0.3621	0.4037	**0.4067**	0.3792	**0.4052**	**0.4052**
PEE	BLEU	0.3500	**0.4003**	0.4001	0.3792	**0.3819**	**0.3819**
F1	F1	0.3499	0.3930	**0.3940**	0.3661	0.3933	**0.3935**
PEE	F1	0.3432	**0.3886**	0.3882	0.3650	**0.3715**	**0.3715**
F1	BLEUs	0.3316	0.3839	**0.3849**	0.3512	0.3848	**0.3851**
PEE	BLEUs	0.3209	**0.3793**	0.3777	0.3503	**0.3585**	**0.3585**
F1	acc	0.6998	0.7296	**0.7351**	**0.8300**	0.7583	0.7660
PEE	acc	0.6623	0.7318	**0.7384**	0.8311	0.8344	**0.8466**
F1	acc-best	0.4724	**0.5000**	0.4989	**0.9724**	0.5828	0.5894
PEE	acc-best	0.3256	**0.5243**	0.5011	**0.9437**	0.9415	**0.9437**

*Table 5. 1-best translation results on DEAMT09 using general or individual SVR models predicting either F_1 or PEE. Top results are in **bold**.*

2.17 F_1 points compared to the top MT system. QuEst is also able to achieve higher accuracy than the previously reported 0.77 (Specia et al., 2010).

3.2. Correlation with Human Judgments

Here we rank the translations according to the predicted scores and evaluate their correlation with the human rankings. Table 6 presents the results for the DEAMT09 dataset using a single general prediction model and individual models for each MT system. The results show that PEE predictions generally correlate better with human judgments than F_1 predictions.

τ	Target	BL	IR	BL+IR
General	F1	0.6732	0.6064	**0.6382**
	PEE	0.6787	0.6124	**0.7034**
Individual	F1	**0.7719**	0.6743	0.6853
	PEE	0.7719	0.7922	**0.8070**

Table 6. Kendall's τ between the predicted ranks and human judgments for DEAMT09.

4. n-Best List Re-ranking and Combination

In this section we describe the use of QuEst to obtain predictions on the quality of translations in n-best lists in order to re-rank these lists to have the best predicted translation ranked first, or combine translations in these lists to generate a new, better translation. Translation quality improvements using re-ranking or combination allow SMT system independent gains. Re-ranking is done at the sentence-level by using quality predictions to rank translations from n-best lists and select the top ones.

	Features	Setting	BLEU	F_1	BLEUs	NISTs	1-WER
		1-best	0.1710	0.1725	0.1334	1.6445	0.1825
Re-ranking	50-best	oracle	0.2033	0.2087	0.1686	1.8168	0.2340
	BL	FS	0.1627	0.1684	0.1222	***1.6486**	0.1069
	IR	FS	**0.1705**	**0.1712**	**0.1323**	1.6414	**0.1739**
	BL+IR		0.1668	0.1696	0.1275	*1.6449	0.1576
	100-best	oracle	0.2105	0.2160	0.1769	1.8479	0.2485
	BL	FS	0.1639	0.1687	0.1233	1.6383	0.0919
	IR	FS	0.1691	0.1692	**0.1309**	1.6368	**0.1714**
	BL+IR	PLS	**0.1696**	**0.1697**	0.1293	**1.6409**	0.1564
Word Combination	250-best list	oracle	0.2196	0.2253	0.1873	1.8816	0.2609
	BL		0.1705	**0.1721**	0.1323	***1.6455**	*0.1850
	IR	FS	0.1703	0.1718	*0.1337	1.6403	*0.1931
	BL+IR		**0.1707**	**0.1721**	***0.1354**	1.6433	***0.1945**
	1000-best	oracle	0.2360	0.2412	0.2052	1.9472	0.2783
	BL	PLS	0.1707	0.1719	*0.1353	1.6328	*0.1949
	IR	PLS	***0.1716**	**0.1723**	*0.1355	1.6363	*0.1965
	BL+IR	PLS	*0.1715	0.1718	***0.1362**	**1.6384**	***0.1992**

*Table 7. DQET13-nbest results. * achieve improvements; top results are in **bold**.*

Combination is done at the word-level by using lattice re-scoring to obtain combined translations from translation hypotheses that minimize overall word error.

Re-ranking results show that we can improve over 1-best results, with 100-best lists leading to the best results. Word-level combination results show that the performance increase as we increase n and the best results are obtained with 1000-best lists where 1000 is the largest n we experimented with. We predict F_1 scores and retain the top results among different settings achieving improvements according to F_1 or overall.

Word-level combination is obtained by converting each n-best list into a word lattice and finding the word-level combination of translation hypotheses that minimizes WER. A word lattice is a partially ordered graph with word hypotheses at the nodes (Mangu et al., 2000). An n-best lattice rescoring (Mangu et al., 2000) functionality is provided by the SRILM (Stolcke, 2002) toolkit. Each hypothesis in a given n-best list is weighted with the predicted scores, converted into a word lattice format, aligned, and the best hypothesis minimizing the WER is selected as the consensus hypothesis. As we see in the results, the word-level combination approach is able to improve the performance more than sentence-level re-ranking due to reasons including (Mangu et al., 2000): (i) lattice representation is able to consider alternative translations, (ii) pruning of the lattices minimizes word errors and leads to better modeling of word posterior probabilities, (iii) WER minimization may be a good target to optimize for translation performance.

	Features	Setting	BLEU	F_1	BLEUs	NISTs	1-WER
English-Spanish		1-best	0.2690	0.2673	0.2228	2.3278	0.3920
	100-best	oracle	0.3560	0.3667	0.3351	2.6493	0.4940
Re-ranking	BL	FS	0.2535	0.2456	0.1947	2.2702	0.3187
	IR	PLS	0.2516	0.2457	0.2010	2.2339	**0.3817**
	SMT	PLS	0.2569	**0.2498**	0.2008	**2.2804**	0.3451
	BL+IR		0.2567	0.2465	0.2011	2.2614	0.3610
	IR+SMT	PLS	**0.2576**	0.2495	**0.2024**	2.2796	0.3574
	BL+IR+SMT		0.2564	0.2482	0.2012	2.2741	0.3585
Word Comb.	BL	FS	0.2676	0.2653	0.2208	**2.3212**	*0.3925
	IR	FS	**0.2682**	**0.2661**	**0.2216**	2.3207	***0.3936**
	SMT		0.2672	0.2648	0.2196	2.3197	*0.3928
	BL+IR	PLS	0.2676	0.2651	0.2204	2.3198	0.3927
	IR+SMT	PLS	0.2677	0.2647	0.2196	2.3179	0.3915
	BL+IR+SMT		0.2677	0.2652	0.2198	2.3195	0.3926
Spanish-English		1-best	0.2816	0.2816	0.2335	2.3696	0.4064
	100-best	oracle	0.3763	0.3902	0.3554	2.6858	0.5154
Re-ranking	BL	PLS	0.2656	**0.2614**	**0.2112**	**2.3441**	0.3663
	IR	FS	0.2646	0.2553	0.2025	2.2888	0.3819
	SMT	FS	0.2602	0.2578	0.2052	2.3094	0.3559
	BL+IR	FS	0.2660	0.2573	0.2051	2.2932	**0.3849**
	IR+SMT	PLS	0.2616	0.2552	0.2031	2.2940	0.3692
	BL+IR+SMT	FS	**0.2662**	0.2572	0.2045	2.2922	0.3827
Word Comb.	BL	PLS	***0.2818**	**0.2803**	**0.2315**	**2.3686**	*0.4124
	IR		***0.2818**	0.2801	0.2314	2.3642	*0.4127
	SMT	PLS	*0.2817	0.2795	0.2301	2.3628	*0.4116
	BL+IR	FS	0.2815	0.2792	0.2298	2.3673	*0.4128
	IR+SMT	FS	0.2815	0.2793	0.2304	2.3652	***0.4131**
	BL+IR+SMT	PLS	0.2816	0.2797	0.2304	2.3636	*0.4127

Table 8. DFDA13-nbest results with 100*-best lists. Top results are in* ***bold****.*

Table 7 presents 1-best translation baseline, oracle translation according to F_1, and the 1-best translation results after sentence-level re-ranking according to the predicted F_1 scores using 50-best or 100-best lists, and after word-level combination using 250-best or 1000-best lists on the DQET13-nbest dataset. QuEst is able to improve the performance on all metrics, with IR or BL+IR feature sets obtain the best results. F_1 is also improved with word-level combination using a 100-best list (Specia et al., 2014). We obtain up to 0.28 points increase in BLEUs with the word-level combination using a 1000-best list. With sentence-level re-ranking, we are able to improve only the NISTs scores using a 50-best list.

With sentence-level re-ranking, we observe that performance decrease as we increase n from 50 to 1000. Results for different n-best lists with increasing n are pre-

sented in (Specia et al., 2014). With word-level combination, performance increase as we increase n on DQET-nbest and on DFDA13-nbest, we observe increasing performance on some metrics with increasing n where the best results are obtained with 100-best lists.

Table 8 presents the corresponding results on the DFDA13-nbest English-Spanish and Spanish-English datasets using 100-best lists. Sentence-level re-ranking does not yield improvements. With word-level combination, we are able to achieve improvements according to BLEU and 1-WER. The performance gains are larger on DFDA13-nbest and the addition of SMT features improve the performance slightly. The IR feature set and other feature sets containing IR lead to the best results. Previous results such as (Blatz et al., 2004) could not improve the BLEU scores with n-best list re-scoring, but obtained some improvements in NISTs scores.

With word-level combination, performance increase as we increase n for NISTs in general and for 1-WER when translating from English to Spanish. We observed a slight decrease in 1-WER when translating from Spanish to English. NIST favors more diverse outputs by weighting less frequent n-grams more, which can explain the increase in NISTs scores with increasing n.

5. ITERPE and SSSS for Machine Translation Improvements

In these experiments we use quality estimation to identify source sentences whose translations have potential for improvement such that building sentence-specific SMT systems (SSSS) may improve their translations and the overall SMT performance. Our goal is to find instances that have suboptimal translations and for which a better translation is possible by building SSSS. In domain adaptation, we show that Moses SMT systems built with FDA-selected $10,000$ training sentences are able to obtain F_1 results as good as the baselines that use up to 2 million sentences and better performance with Moses SMT systems built with FDA-selected $50,000$ training sentences (Biçici, 2015). In fact, SSSS built using as few as $5,000$ training instances for each source sentence can achieve close performance to a baseline SMT system using 2 million sentences (Biçici, 2011). In Table 14, we show that we can achieve better NIST performance using SSSS built with FDA5-selected $5,000$ training instances. The ITERPE model allows us to identify which sentences have more potential for improvement by re-translation and we demonstrate performance improvements with SSSS. An ITERPE sorting of the translation instances can be used to group them into different quality bands for different purposes, for instance, for re-translation or for post-editing.

5.1. ITERPE: Identifying Translation Errors Regardless of Prediction Errors

We use the IR feature set to build two QuEst systems: QuEst_S and $\text{QuEst}_{(S,T')}$, where the former uses only the source sentence S and the latter uses both S and its translation T' when predicting the quality score. $\text{QuEst}_{(S,T')}$ is a more informed pre-

	Dataset	BLEU	F_1	$\hat{y}$	$\hat{y}_S$	$\hat{y} - \hat{y}_S$	$\lvert\hat{y} - \hat{y}_S\rvert$	$\bar{F}_{1S} - F_1$	$\bar{F}_{1T} - F_1$
en-de	DFDA14-train	0.2223	0.2360	0.2283	0.2436	-0.0153	0.1069	0.1734	0.1554
	DFDA14-test	0.1761	0.2093	0.2147	0.1800	0.0347	0.0738	0.1194	0.1545
de-en	DFDA14-train	0.2570	0.2580	0.2578	0.2649	-0.0071	0.0096	0.1094	0.1478
	DFDA14-test	0.2410	0.2580	0.2529	0.2663	-0.0259	0.1535	0.0400	0.1248

Table 9. Training and testing average statistics and ITERPE results. y *is* F_1.

dictor. Biçici et al. (2013) obtained better results than QuEst using only the source sentences with the machine translation performance prediction system.

We consider two types of errors: prediction error or translation error. Prediction errors are errors due to the prediction model, while translation errors are errors due to mistranslations by the MT engine. We want to fix translation errors regardless of potential prediction errors. Having a precise estimator (low MAE) is important for identifying the score differences. Also, if the prediction reaches the top possible target score, topScore, where $y \leq$ topScore then we do not expect to be able to further improve the translation. Let $\hat{y}_S = \text{QuEst}_S(S)$ and $\hat{y} = \text{QuEst}_{(S,T')}(S, T')$ represent the prediction from QuEst using only the source S and using both S and T' and t_y be a positive threshold on the prediction. When predicting which instances to re-translate, we compare two strategies, which sort instances according to $\mathbf{d}$:

MEAN: $\mathbf{d} = \bar{\hat{\mathbf{y}}} - \hat{\mathbf{y}}$. Selects instances whose expected performance is lower than the expected mean performance, which attempts to improve lower performing instances.

ITERPE: $\mathbf{d} = \hat{\mathbf{y}} - \hat{\mathbf{y}}_S$. Selects instances according to differences in predictions from different predictors, which attempts to identify the translation errors regardless of prediction errors (ITERPE).

The ITERPE strategy relies on the performance prediction of the quality of a sentence translation task by two separate predictors, one using only the source sentence and one using both the source sentence and the translation. ITERPE invention (Biçici, 2014) works as follows:

- If $\hat{y}_S > \hat{y}$ and $\hat{y}_S <$ topScore, then by looking at S, we expect a better translation performance. So, T' is not optimal.
- If $\hat{y}_S = \hat{y}$, then either the quality score is the same for both or T' is not giving us new information. If $(\hat{y} - \hat{y}_S) \leq t_y$ and $\hat{y}_S <$ topScore, then we assume that T' is not optimal.
- If $(\hat{y} - \hat{y}_S) > t_y$, then T' may be close to the possible translation we can obtain.

5.2. ITERPE Learning Results

t_y can be optimized to improve the overall F_1 performance on the training set. Table 9 shows the average English to German (en-de) and German to English (de-en)

	$\hat{y} - \hat{y}_S \leq$	n	y	MAE	MAE_S	MAER	$MAER_S$	MRAER	$MRAER_S$	$\lceil y_S \rceil - y$	$\lceil y_T \rceil - y$
en-de	-0.1642	86	0.0893	0.0356	0.1517	0.3688	2.0522	0.3306	1.0422	0.2467	0.2533
	-0.0153	1920	0.1726	0.0639	0.1123	0.3624	0.8543	0.4582	0.8612	0.199	0.1935
	0.0	123	0.2206	0.0841	0.0866	0.4393	0.4636	0.5577	0.5816	0.1578	0.1531
	0.1335	434	0.2751	0.0636	0.0731	0.2623	0.263	0.3745	0.4361	0.1246	0.1077
	0.2823	130	0.4382	0.0613	0.1973	0.1598	0.4306	0.2433	0.7727	0.1122	0.0067
	0.4311	102	0.6125	0.0579	0.3607	0.0912	0.5781	0.1372	0.9219	0.1188	-0.0274
	0.5799	71	0.8007	0.0826	0.5447	0.0971	0.6757	0.1357	0.9633	-0.0054	-0.0948
	1.0	21	0.9272	0.0683	0.6713	0.0743	0.7223	0.1006	0.9721	-0.1211	-0.0841
de-en	-0.1422	228	0.1057	0.034	0.1563	0.3457	1.8446	0.3097	1.0179	0.2026	0.2645
	-0.0071	1781	0.2076	0.0719	0.1125	0.3373	0.6719	0.4556	0.7979	0.1404	0.1843
	0.0	54	0.2357	0.0527	0.0544	0.2525	0.2627	0.4006	0.4155	0.1099	0.1608
	0.128	565	0.2871	0.0484	0.0588	0.199	0.2079	0.3018	0.3713	0.0621	0.1085
	0.2631	183	0.4328	0.0514	0.1736	0.1345	0.3869	0.2118	0.7291	0.008	0.0002
	0.3983	99	0.6079	0.06	0.341	0.0981	0.5514	0.1574	0.9173	-0.0177	-0.0778
	0.5334	56	0.7869	0.0889	0.5148	0.1163	0.6417	0.1593	0.9521	-0.1157	-0.123
	1.0	20	0.9064	0.094	0.636	0.103	0.6996	0.1442	0.9805	-0.1924	-0.1686

Table 10. DFDA14-train instances binned according to their deviation from $\bar{\mathbf{d}}$ using ITERPE. y is F_1.

training set and test set statistics. In this case, the target, y, is the F_1 score. MAE is for $\hat{y}$ and MAE_S is for $\hat{y}_S$. $\lceil y_S \rceil$ and $\lceil y_T \rceil$ are the source and target performance bounds on y calculated based on synthetic translations (Biçici et al., 2013). Score differences to bounds show a measure of how close are the translations to the bounds.

We cumulatively and separately bin instances according to their deviation from the mean of $\mathbf{d}$, $\bar{\mathbf{d}}$, in $\sigma_{\mathbf{d}}$ steps and evaluate the prediction performance of different strategies. Table 11 shows the cumulatively binning performance on the training set where predictions are obtained by cross-validation and Table 10 show the training results within separate bins. n is the number of instances in the score range. Table 11 also shows that ITERPE is able to identify hard to translate instances using only the prediction information.

Table 12 and Table 13 show the cumulatively binning of the performance of the test set instances according to their deviation from $\bar{\mathbf{d}}$ for both the en-de and de-en DFDA14-test sets. We observe that lower $\mathbf{d}$ corresponds to instances with lower F_1 scores (compared to $\bar{\mathbf{y}}$) and higher potential for improvements according to $\lceil y_S \rceil - y$ and $\lceil y_T \rceil - y$. Until about $d \leq \bar{\mathbf{d}} + 2\sigma_{\mathbf{d}}$, MAE_S is lower, which indicates that for these instances, we can trust $\hat{y}_S$ more than $\hat{y}$ and therefore, since $\hat{y}_S > \hat{y}$ when $d < 0$, these instances correspond to the instances that have some potential for improvement. Including the confidence of predictions in the decision process may also improve the performance.

	$\mathbf{d} \leq$	n	$\bar{y}$	MAE	MAE_S	MAER	$MAER_S$	MRAER	$MRAER_S$	$\lceil y_S \rceil - y$	$\lceil y_T \rceil - y$
en-de	-0.1642	86	0.0893	0.0356	0.1517	0.3688	2.0522	0.3306	1.0422	0.2467	0.2533
	-0.0153	2006	0.1691	0.0627	0.1139	0.3626	0.9056	0.4527	0.869	0.2011	0.1961
	0.0	2129	0.172	0.0639	0.1124	0.3671	0.8801	0.4588	0.8524	0.1986	0.1936
	0.1335	2563	0.1895	0.0639	0.1057	0.3493	0.7756	0.4445	0.7819	0.186	0.179
	0.2823	2693	0.2015	0.0638	0.1101	0.3402	0.759	0.4348	0.7815	0.1825	0.1707
	0.4311	2795	0.2165	0.0635	0.1193	0.3311	0.7524	0.4239	0.7866	0.1802	0.1635
	0.5799	2866	0.231	0.064	0.1298	0.3253	0.7505	0.4168	0.791	0.1756	0.1571
	1.0	2887	0.236	0.064	0.1338	0.3235	0.7502	0.4145	0.7923	0.1734	0.1553
de-en	-0.1422	228	0.1057	0.034	0.1563	0.3457	1.8446	0.3097	1.0179	0.2026	0.2645
	-0.0071	2009	0.196	0.0676	0.1175	0.3383	0.805	0.4391	0.8229	0.1475	0.1934
	0.0	2063	0.1971	0.0672	0.1158	0.336	0.7908	0.438	0.8122	0.1465	0.1925
	0.128	2628	0.2164	0.0632	0.1036	0.3066	0.6655	0.4087	0.7174	0.1283	0.1745
	0.2631	2811	0.2305	0.0624	0.1081	0.2954	0.6473	0.3959	0.7182	0.1205	0.1631
	0.3983	2910	0.2433	0.0623	0.116	0.2887	0.6441	0.3878	0.725	0.1158	0.1549
	0.5334	2966	0.2536	0.0628	0.1236	0.2854	0.644	0.3835	0.7292	0.1114	0.1497
	1.0	2986	0.258	0.063	0.127	0.2842	0.6444	0.3819	0.7309	0.1094	0.1475

Table 11. DFDA14-train instances cumulatively binned based on deviation from $\bar{\mathbf{d}}$ using ITERPE. y is F_1.

	$\hat{y} - \hat{y}_S \leq$	n	y	MAE	MAE_S	MAER	$MAER_S$	MRAER	$MRAER_S$	$\lceil y_S \rceil - y$	$\lceil y_T \rceil - y$
MEAN	-0.1035	91	0.192	0.1106	0.1202	0.475	0.9383	1.0233	0.9838	0.1738	0.2155
	0.0	1792	0.2013	0.0854	0.1025	0.5055	0.7783	0.8717	1.1362	0.1256	0.1522
	0.0	1792	0.2013	0.0854	0.1025	0.5055	0.7783	0.8717	1.1362	0.1256	0.1522
	0.1035	2449	0.2086	0.094	0.1064	0.5802	0.7722	0.9338	1.1267	0.1172	0.1503
	0.2069	2609	0.2095	0.0992	0.1065	0.6291	0.7691	1.0152	1.1211	0.1175	0.1513
	0.3104	2673	0.2094	0.1038	0.107	0.6732	0.7723	1.0691	1.1197	0.1185	0.1524
	0.4139	2703	0.2093	0.107	0.1071	0.7021	0.7738	1.1087	1.1184	0.119	0.1535
	1.0	2737	0.2093	0.1125	0.1075	0.747	0.7764	1.173	1.119	0.1195	0.1548
ITERPE	-0.129	54	0.2217	0.1215	0.1294	0.4418	0.8893	1.0072	1.1438	0.1209	0.1647
	-0.0202	1874	0.2036	0.0856	0.1041	0.4963	0.7788	0.8894	1.1932	0.1168	0.1467
	0.0	2053	0.2046	0.0878	0.105	0.5225	0.7846	0.8941	1.1761	0.1166	0.1477
	0.0885	2438	0.2083	0.0942	0.1065	0.5791	0.7737	0.9385	1.1394	0.1161	0.1497
	0.1972	2606	0.2093	0.0992	0.1065	0.6257	0.767	1.0161	1.1285	0.117	0.1513
	0.306	2671	0.2092	0.1036	0.1067	0.6682	0.7696	1.0733	1.1276	0.1181	0.1523
	0.4147	2704	0.2093	0.1071	0.1072	0.6998	0.7715	1.1057	1.1252	0.1186	0.1534
	1.0	2737	0.2093	0.1125	0.1075	0.7414	0.7728	1.1735	1.1258	0.1192	0.1546

Table 12. DFDA14-test instances cumulatively binned based on deviation from $\bar{\mathbf{d}}$ for en-de comparing different strategies. y is F_1.

	$\hat{y} - \hat{y}_S \leq$	n	y	MAE	MAE_S	MAER	$MAER_S$	MRAER	$MRAER_S$	$\lceil y_S \rceil - y$	$\lceil y_T \rceil - y$
MEAN	-0.1065	87	0.2668	0.1552	0.1299	0.4913	0.663	1.1651	0.8625	0.0633	0.1363
	-0.0	2090	0.2553	0.1023	0.1033	0.4303	0.5538	0.9059	0.9115	0.0344	0.1146
	0.0	2090	0.2553	0.1023	0.1033	0.4303	0.5538	0.9059	0.9115	0.0344	0.1146
	0.1065	2720	0.2559	0.1053	0.1041	0.4863	0.56	0.9321	0.9067	0.0383	0.1205
	0.2131	2860	0.2559	0.1094	0.1044	0.5201	0.5605	0.9921	0.9077	0.0396	0.1222
	0.3196	2922	0.2567	0.1122	0.1046	0.5379	0.5585	1.0357	0.906	0.0397	0.1226
	0.4262	2963	0.257	0.1155	0.105	0.561	0.5605	1.0695	0.9062	0.0402	0.1238
	1.0	3003	0.258	0.1205	0.1059	0.5896	0.561	1.1067	0.9045	0.04	0.1247
ITERPE	-0.126	114	0.2829	0.1408	0.1154	0.4528	0.6344	1.2341	1.0201	-0.0163	0.0774
	-0.014	2019	0.2565	0.1023	0.1035	0.4232	0.5549	0.9101	0.9206	0.0314	0.1121
	0.0	2204	0.2564	0.1029	0.1042	0.4343	0.5569	0.9049	0.9165	0.0324	0.1137
	0.098	2708	0.2558	0.1052	0.1042	0.4851	0.5615	0.9325	0.9092	0.0384	0.1202
	0.2101	2863	0.2561	0.1094	0.1045	0.5183	0.5603	0.9956	0.9085	0.0395	0.1221
	0.3221	2930	0.2567	0.1129	0.1047	0.5437	0.5605	1.041	0.906	0.0399	0.1228
	0.4341	2966	0.2569	0.116	0.1049	0.5645	0.561	1.0785	0.9058	0.0404	0.1239
	1.0	3003	0.258	0.1205	0.1059	0.59	0.5616	1.1097	0.9044	0.0401	0.1247

Table 13. DFDA14-test instances cumulatively binned based on deviation from $\bar{\boldsymbol{d}}$ for de-en comparing different strategies. y is F_1.

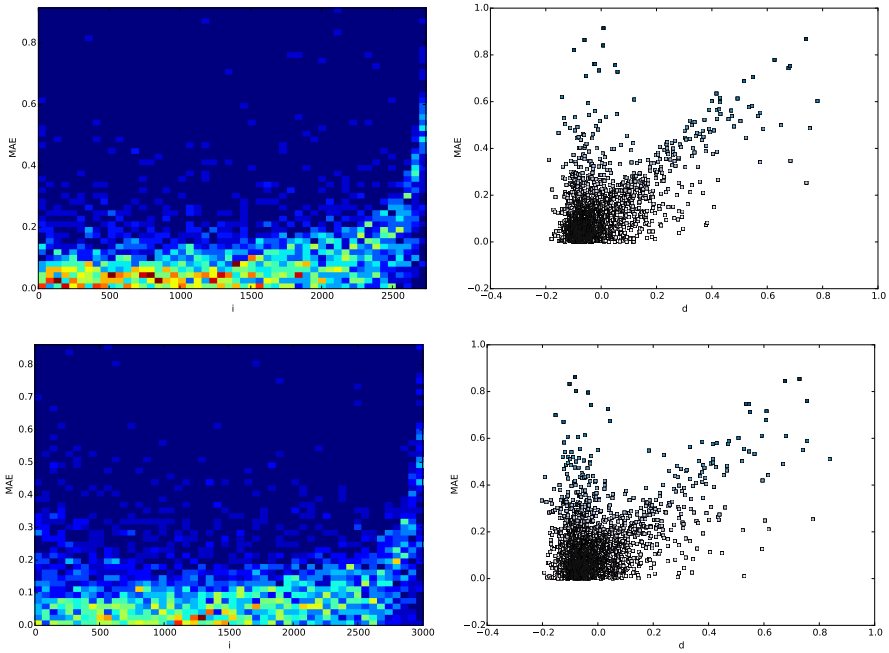


*Figure 1. ITERPE sorted MAE histogram and **d** vs. MAE plot for en-de (top) and de-en (bottom) on the DFDA14-test test set. The increase in MAE is visible.*

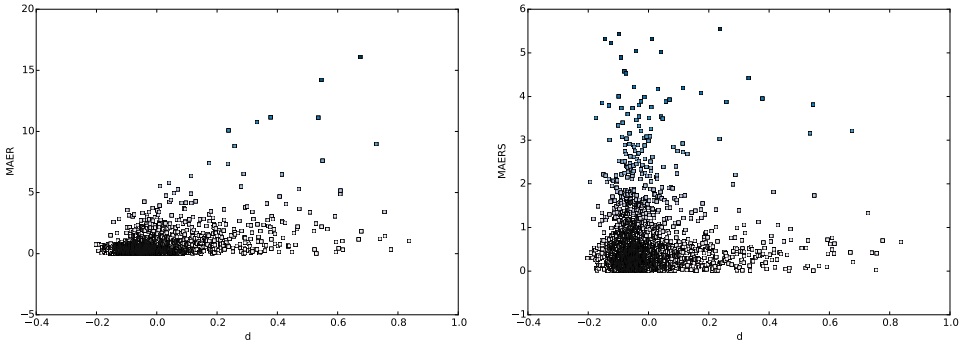


Figure 2. ITERPE sorted MAER vs. $MAER_S$ plot for de-en test set.

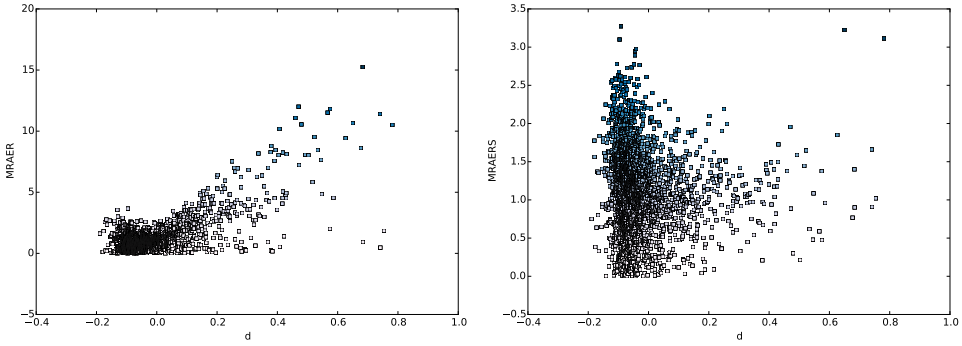


Figure 3. ITERPE sorted MRAER vs. $MRAER_S$ plot for en-de test set.

We observe that the difference with the upper bounds increase with decreasing **d**, which indicates that we are able to identify instances that have the highest potential for increase in their performance. ITERPE is able to consistently identify instances with higher potential difference to the bounds. Figure 1 plots ITERPE sorted MAE histogram and **d** vs. MAE plot for en-de and de-en on the DFDA14-test test set.

Figure 2 compares the MAER and $MAER_S$ distributions on de-en DFDA14-test test set instances sorted according to ITERPE and Figure 3 compares the MRAER and $MRAER_S$ distributions on en-de DFDA14-test test set.

5.3. SSSS with ITERPE Sorted Instances

We build SSSS over the en-de and de-en DFDA14-test set instances using up to 25,000 training instances selected specifically for each source sentence using FDA5 (Biçici and Yuret, 2015). The results are presented in Table 14, which show that using training set of 1,000 instances for building SSSS does not lead to better results on all of DFDA14-test. However, SSSS with 1,000 training instances each can obtain as close results as 1 F_1 point in en-de and shows that this level of parallelism is possible with SMT. en-de translation performance also improves with SSSS where each use 5,000 training instances.

We also run sentence-specific SMT experiments over the top instances that have more potential for improvement according to their ITERPE sorting. Table 14 also presents the results for the top 100 or 200 instances compared to baseline translation performance on those instances. Improvements of up to 1.43 BLEU, 0.54 F_1, 2.9 NIST, 0.64 BLEUs, and 4.7 NISTs points are obtained over the top 100 ITERPE sorted instances for en-de. Compared to baseline results, ITERPE sorting is able to obtain up to 1.43 BLEU points improvement over the top 100 instances by being able to identify the top instances that have more potential for improvement by re-translation with SSSS.

Figure 4 plots ITERPE sorted accumulative average performance improvement compared with the baseline over the top 500 instances in en-de and de-en on the DFDA14-test test set. Maximum accumulative gain with SSSS on the DFDA14-test test set over the top 100 ITERPE sorted instances is visible in Figure 4 and can reach 4.4 BLEUs points and 3.7 F_1 points for en-de, and 9.1 BLEUs points and 7.2 F_1 points for de-en. Maximum instance gains with SSSS on the DFDA14-test test set over the top 100 ITERPE sorted instances are presented in Table 15.

6. Conclusions

We described several ways to use quality predictions produced by the QuEst framework in order to improve SMT performance without changing the MT systems' internal functioning. In all cases, promising results were found, leading us to believe that quality predictions produced by the state of the art approaches can help in the process of achieving the high quality translation goal. Table 16 summarizes our main findings when we use quality predictions towards improving machine translation quality.

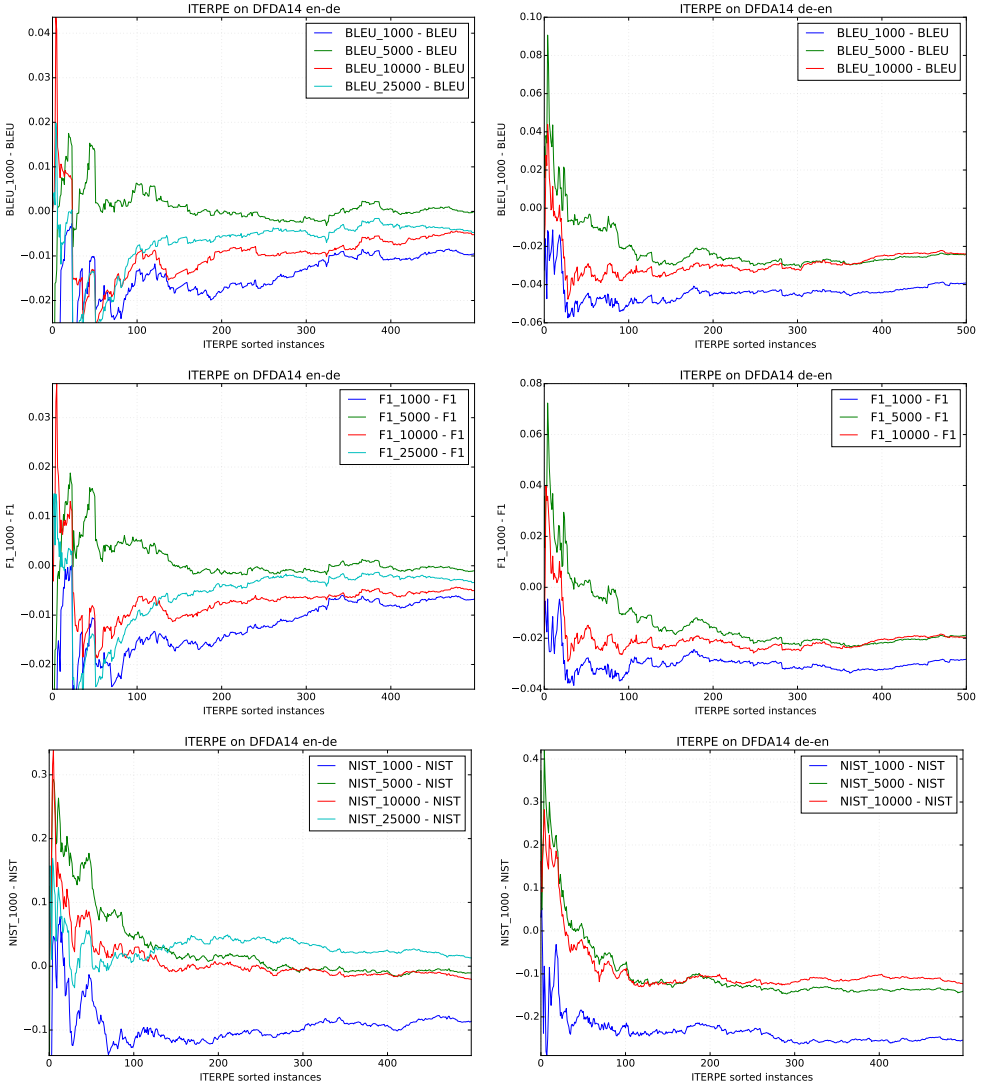


Figure 4. ITERPE sorted accumulative average F_1*, BLEUs, or NISTs performance improvement compared with the baseline over the top* 500 *instances in en-de (left) and de-en (right) on the DFDA14-test test set.*

	n	# train	Baseline BLEU	F_1	NIST	BLEUs	NISTs	ITERPE + SSSS BLEU	F_1	NIST	BLEUs	NISTs
en-de	100	1000	0.1614	0.2048	4.6227	0.1406	1.9119	0.1609	0.1893	4.6152	0.1271	1.7847
		5000						**0.1757**	**0.2102**	**4.9111**	**0.147**	**1.9587**
		10000						**0.1633**	0.1974	**4.8022**	0.1309	**1.9314**
		25000						**0.1647**	0.1942	**4.7821**	0.1323	**1.9192**
	200	1000	0.1569	0.1952	4.9408	0.1284	1.9558	0.1444	0.18	4.9064	0.1102	1.8408
		5000						**0.1577**	0.1937	**5.1326**	0.1278	**1.9715**
		10000						0.1542	0.1872	**5.0878**	0.1191	**1.9587**
		25000						**0.1574**	0.1909	**5.1462**	0.1226	**1.9982**
	2737	1000	0.1761	0.2093	5.9602	0.1435	2.0366	0.1632	0.199	5.9596	0.1295	1.9277
	2737	5000						0.1725	0.2035	**6.072**	0.1378	1.9821
	1500	10000	0.1726	0.2017	5.8253	0.1373	2.1071	0.1678	0.1958	**5.9166**	0.1314	2.0702
	750	25000	0.1741	0.1998	5.6429	0.1364	2.0981	**0.1747**	0.198	**5.7945**	0.1341	**2.109**
de-en	100	1000	0.2831	0.2842	5.8797	0.2446	2.5136	0.2402	0.2509	5.3697	0.1935	2.2959
		5000						0.2672	0.2749	5.7442	0.2246	2.4365
		10000						0.2602	0.2611	5.7401	0.2103	2.4243
	200	1000	0.2849	0.2918	6.337	0.2476	2.5262	0.2421	0.263	5.7436	0.2022	2.3065
		5000						0.2625	0.2759	6.1281	0.2234	2.4169
		10000						0.2599	0.2703	6.1347	0.2168	2.4199
	3003	1000	0.241	0.258	7.2325	0.198	2.3194	0.2007	0.233	6.4235	0.1663	2.0358
	1250	5000	0.2516	0.2623	7.0413	0.2095	2.4641	0.2269	0.2465	6.7374	0.1897	2.3049
	700	10000	0.267	0.2752	6.8915	0.2269	2.5233	0.2438	0.2592	6.6454	0.2072	2.3977

Table 14. ITERPE + SSSS results on DFDA14-test over the top n ITERPE sorted instances. We build SSSS for each instance.

	# train	ITERPE + SSSS BLEUs	F_1	NISTs
en-de	1000	24.6	18.8	78.4
	5000	27.7	21.4	105.8
	10000	21.1	18.9	126.4
	25000	49.3	44.9	120.3
de-en	1000	31.4	26.9	99.8
	5000	54.2	51.1	124.2
	10000	32.0	27.1	113.3

Table 15. Maximum instance improvement points with ITERPE + SSSS on DFDA14-test test set over the top 100 ITERPE sorted instances.

Improvement Points	BLEU	F_1	NIST	BLEUs	NISTs
multiple system translation ranking	2.72	2.17		3.66	
n-best list re-ranking					0.41
n-best list combination	0.06			0.28	0.31
ITERPE (en-de, $n = 100$)	1.43	0.54	2.9	0.64	4.7

Table 16. Summary of improvement points over the baseline obtained using QuEst *for high quality machine translation.*

The ITERPE model can obtain robust sortings of the translations allowing us to answer questions about which translations do not have much potential for improvement and which may need to be re-translated or maybe post-edited. We build sentence specific SMT systems on the ITERPE sorted instances identified as having more potential for improvement and obtain improvements in BLEU, F_1, NIST, BLEUs, and NISTs.

Acknowledgements

This work is supported in part by SFI as part of the ADAPT CNGL Centre for Global Intelligent Content (`http://www.adaptcentre.ie`, 07/CE/I1142) at Dublin City University, in part by SFI for the project "Monolingual and Bilingual Text Quality Judgments with Translation Performance Prediction" (`computing.dcu.ie/~ebicici/Projects/TIDA_RTM.html`, 13/TIDA/I2740), and in part by the European Commission through the QTLaunchPad FP7 project (`http://www.qt21.eu`, No: 296347). We also thank the SFI/HEA Irish Centre for High-End Computing (ICHEC, `http://www.ichec.ie`) for the provision of computational facilities and support.

Bibliography

Biçici, Ergun. *The Regression Model of Machine Translation*. PhD thesis, Koç University, 2011. Supervisor: Deniz Yuret.

Biçici, Ergun. Feature decay algorithms for fast deployment of accurate statistical machine translation systems. In *Proc. of the Eighth Workshop on Statistical Machine Translation*, Sofia, Bulgaria, August 2013a. Association for Computational Linguistics.

Biçici, Ergun. Referential translation machines for quality estimation. In *Proc. of the Eighth Workshop on Statistical Machine Translation*, pages 343–351, Sofia, Bulgaria, August 2013b. Association for Computational Linguistics.

Biçici, Ergun. ITERPE: Identifying translation errors regardless of prediction errors, 2014. Invention Disclosure, DCU Invent Innovation and Enterprise `https://www.dcu.ie/invent/`. USPTO filing number: US62093483.

Biçici, Ergun. Domain adaptation for machine translation with instance selection. *The Prague Bulletin of Mathematical Linguistics*, 103, 2015.

Biçici, Ergun and Deniz Yuret. RegMT system for machine translation, system combination, and evaluation. In *Proc. of the Sixth Workshop on Statistical Machine Translation*, pages 323–329, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W11-2137`.

Biçici, Ergun and Deniz Yuret. Optimizing instance selection for statistical machine translation with feature decay algorithms. *IEEE/ACM Transactions On Audio, Speech, and Language Processing (TASLP)*, 23:339–350, 2015. doi: 10.1109/TASLP.2014.2381882.

Biçici, Ergun, Declan Groves, and Josef van Genabith. Predicting sentence translation quality using extrinsic and language independent features. *Machine Translation*, 27:171–192, December 2013. ISSN 0922-6567. doi: 10.1007/s10590-013-9138-4.

Biçici, Ergun, Qun Liu, and Andy Way. Parallel FDA5 for fast deployment of accurate statistical machine translation systems. In *Proc. of the Ninth Workshop on Statistical Machine Translation*, pages 59–65, Baltimore, USA, June 2014. Association for Computational Linguistics.

Björnsson, Carl Hugo. *Läsbarhet*. Liber, 1968.

Blatz, John, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. Confidence estimation for machine translation. Technical report, 2004.

Bojar, Ondřej, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proc. of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-2201`.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, and Omer F. Zaidan. Findings of the 2011 Workshop on Statistical Machine Translation. In *Proc. of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, England, July 2011. Association for Computational Linguistics.

Doddington, George. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. of the second international conference on Human Language Technology Research*, pages 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

Guyon, Isabelle, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422, 2002.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *ACL*, pages 177–180, Prague, Czech Republic, June 2007.

Mangu, Lidia, Eric Brill, and Andreas Stolcke. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400, 2000.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proc. of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002.

Smola, Alex J. and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, Aug. 2004. ISSN 0960-3174.

Specia, Lucia, Nicola Cancedda, Marc Dymetman, Marco Turchi, and Nello Cristianini. Estimating the sentence-level quality of machine translation systems. In *Proc. of the 13th Annual Conference of the European Association for Machine Translation (EAMT)*, pages 28–35, Barcelona, Spain, May 2009.

Specia, Lucia, Dhwaj Raj, and Marco Turchi. Machine translation evaluation versus quality estimation. *Machine Translation*, 24(1):39–50, 2010. ISSN 0922-6567. doi: 10.1007/s10590-010-9077-2. URL http://dx.doi.org/10.1007/s10590-010-9077-2.

Specia, Lucia, Kashif Shah, Eleftherios Avramidis, and Ergun Biçici. QTLaunchPad deliverable D2.1.3 quality estimation for dissemination, 2013. URL http://www.qt21.eu/launchpad/deliverable/quality-estimation-dissemination.

Specia, Lucia, Kashif Shah, Eleftherios Avramidis, and Ergun Biçici. QTLaunchPad deliverable D2.2.1 quality estimation for system selection and combination, 2014. URL http://www.qt21.eu/launchpad/deliverable/quality-estimation-system-selection-and-combination.

Stolcke, Andreas. Srilm - an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, pages 901–904, 2002.

The Apache Software Foundation. Lucene, 2014. URL http://lucene.apache.org/.

Wikipedia. LIX, 2013. http://en.wikipedia.org/wiki/LIX.

Address for correspondence:
Ergun Biçici
ergun.bicici@computing.dcu.ie
ADAPT CNGL Centre for Global Intelligent Content
School of Computing
Dublin City University
Dublin 9, Dublin, Ireland

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015 65–84

Scalable Reordering Models for SMT based on Multiclass SVM

Abdullah Alrajeh[a b], Mahesan Niranjan[a]

[a] School of Electronics and Computer Science, University of Southampton
[b] Computer Research Institute, King Abdulaziz City for Science and Technology (KACST)

Abstract

In state-of-the-art phrase-based statistical machine translation systems, modelling phrase reorderings is an important need to enhance naturalness of the translated outputs, particularly when the grammatical structures of the language pairs differ significantly. Posing phrase movements as a classification problem, we exploit recent developments in solving large-scale multiclass support vector machines. Using dual coordinate descent methods for learning, we provide a mechanism to shrink the amount of training data required for each iteration. Hence, we produce significant computational saving while preserving the accuracy of the models. Our approach is a couple of times faster than maximum entropy approach and more memory-efficient (50% reduction). Experiments were carried out on an Arabic-English corpus with more than a quarter of a billion words. We achieve BLEU score improvements on top of a strong baseline system with sparse reordering features.

1. Introduction

The mathematical basis of statistical machine translation (SMT) has its origins in the formulation due to Brown et al. (1988), who later introduced five statistical models widely known as the IBM models (Brown et al., 1993). While these early models were word-based, assuming the translation to take place on a word by word basis, in reality, groups of words (phrases) are recognised as better units of translation (Koehn, 2010).

Working at the phrase level helps resolve many ambiguities that occur at the word level. Since the IBM models allow one to many mappings of words, phrase can be automatically defined by training IBM word alignment models in both direction of source and target languages, and combining the two alignments (Och and Ney, 2004).

 Corresponding author: asar1a10@ecs.soton.ac.uk
Cite as: Abdullah Alrajeh, Mahesan Niranjan. Scalable Reordering Models for SMT based on Multiclass SVM. The Prague Bulletin of Mathematical Linguistics No. 103, 2015, pp. 65–84. doi: 10.1515/pralin-2015-0004.

While such attempts at phrase level translation has shown improvement in translation performance, a further issue that has to be addressed is that of long range phrase reorderings (Galley and Manning, 2008). Such reorderings arise from differences in grammatical structures between language pairs and addressing this is important in achieving increased naturalness of the translated output (Koehn, 2010). This issue is particularly pronounced when language pairs separated by large evolutionary distances, or from different linguistic families, are considered such as Arabic and English.

Early work on handling phrase reorderings implemented a relaxation into the decoder which, instead of forcing phrases to be in synchrony, allowed a penalty function that penalised large movements proportionately (Koehn, 2004a). An alternative approach, adopted by several systems nowadays is lexicalised reordering modelling (Tillmann, 2004; Kumar and Byrne, 2005; Koehn et al., 2005), whereby the frequencies of relative positions of the phrase pairs are extracted from the training corpus and used as additional inputs to the decoder (see section 4).

Building on this, some researchers have borrowed powerful ideas from the machine learning literature, to pose the phrase movement problem as a prediction problem using contextual input features whose importance is modelled as weights of a linear classifier trained by entropic criteria. This maximum entropy-based approach (so called MaxEnt) is a popular choice (Zens and Ney, 2006; Xiong et al., 2006; Nguyen et al., 2009; Xiang et al., 2011).

However, if the underlying classification problem is not linearly separable, the MaxEnt classifier will not perform well and more advanced nonlinear methods will be needed. Kernel methods (such as support vector machines in the context of pattern recognition) are state-of-the-art approaches to capture nonlinear effects in datasets (Cristianini and Shawe-Taylor, 2000). They map the data into high dimensional spaces implicitly defined by properties of the chosen kernel, and achieve linear separability in the transformed space.

In many natural language processing problems, including the phrase reordering problem we address here, context information extracted from data are represented explicitly in very high dimensional spaces and linear separability in these spaces can be expected. Motivated by this, Ni et al. (2011) proposed the use of a structured perceptron approach to tackle long range phrase reorderings. While that system results in encouraging results on a Chinese-English translation task, dimensionality and the resulting computational complexity were noted as issues that needed to be tackled.

More recently, there have been extensive developments in the machine learning literature on scaling up support vector machines to problems with large data sizes. The underlying quadratic programming problem is being solved by stochastic gradient search type algorithms. Many researchers proposed fast learning techniques for linear SVM using a dual coordinate descent approach (Hsieh et al., 2008; Glasmachers and Dogan, 2013; Alrajeh et al., 2015). The method of Hsieh et al. (2008), for example, has linear complexity and reaches an ϵ-accurate solution in $O(\log(1/\epsilon))$ iterations. Later, Chang et al. (2010) took the approach a step further and applied linear SVM to

the explicit form of low-degree polynomial kernel. Although, in many cases, kernel mapping is exponential to the input space or infinite as in the Gaussian kernel, the approach is shown to be useful for certain datasets such as NLP task on dependency parsing (Chang et al., 2010).

In this paper, we explore computationally fast and memory-efficient uses of multiclass SVM classifier as a model of long range phrase reorderings. Our results show significant improvement in the BLEU score over a lexicalised reordering model. Training multiclass SVM is shown to be faster than MaxEnt with 50% reduction in memory usage due to a shrinking heuristic we propose.

The remainder of this paper is organised as follows. Section 2 discusses previous work in the field and how it relates to our reordering model. Section 3 gives an overview of the baseline translation system. Section 4 and 5 briefly describe the lexicalised and maximum entropy-based reordering models. Section 6 introduces the proposed SVM-based reordering model. Starting from a brief introduction to the SVM formulation, we explain a fast learning technique for linear mulitclass SVM and how it is extended to nonlinear using kernel mapping. Section 7 evaluates multiclass SVM on benchmark datasets. Section 8 undertakes a comparison between our work and previously proposed models and reports the results evaluated as classification and translation problems. The experiments are based on a large-scale Arabic-English corpus. Finally, we end the paper with a summary of our conclusions and perspectives.

2. Related Work

Adding a lexicalised reordering model has been shown to consistently improve the translation quality for several language pairs (Koehn et al., 2005). The model tries to predict the orientation of a phrase pair with respect to the previous adjacent target words. Ideally, the reordering model would predict the right position in the target sentence given a source phrase, which is difficult to achieve. Therefore, positions are grouped into limited orientations or classes. The orientation probability for a phrase pair is simply based on the relative occurrences in the training corpus.

The lexicalised reordering model has been extended to tackle long-distance reorderings (Galley and Manning, 2008). This takes into account the hierarchical structure of the sentence when considering such an orientation. This approach is shown to improve translation performance for several translation tasks. An additional appeal of the method is the low computing cost.

In addition to the fact that the lexicalised reordering model is always biased toward the most frequent orientation for such a phrase pair, it may suffer from a data sparseness problem since many phrase pairs occur only once (Nguyen et al., 2009). Moreover, the context of a phrase might affect its orientation, which is not considered as well.

Adopting the idea of predicting orientation based on content, it has been proposed to represent each phrase pair by linguistic features as reordering evidence, and then

train a classifier for prediction. The maximum entropy classifier is a popular choice among many researchers.

Zens and Ney (2006) introduced the maximum entropy classifier for phrase reordering. Three different translation tasks were carried out: Arabic-English, Chinese-English and Japanese-English. Only two orientations were considered, left or right (i.e. monotone or swap). Although the proposed model outperforms the relative frequency model in terms of classification performance, they did not draw comparison between them in terms of translation performance. The translation results reported were between their model and the distance-based reordering model. We believe that such a comparison with a lexicalised reordering model is important because the model is faster to estimate (i.e. relative frequency) and also faster to use during translation since there is no overhead computation (i.e. retrieving probabilities from a table).

Xiong et al. (2006) also proposed a maximum entropy model to predicate reordering of neighbour blocks (i.e. phrase pairs) and considered straight or inverted orientations (i.e. monotone or swap). Their experiments were carried out on Chinese-English translation tasks. The reported results were only in terms of translation performance. Similar to Zens and Ney (2006), the authors compared their model with the distance-based reordering model although they did make reference to the lexicalised reordering model.

Nguyen et al. (2009) applied the maximum entropy model to learn orientations identified by the hierarchical reordering model proposed by Galley and Manning (2008). The previous work of Zens and Ney (2006) and Xiong et al. (2006) identified such an orientation without considering the hierarchical structure of previous phrases. The authors used a relatively small English-Vietnamese corpus (0.6 million words) collected from daily newspapers. The approach achieves translation improvements over the lexical hierarchical reordering model in a test set taken from the same corpus (i.e. not a benchmark).

Xiang et al. (2011) introduced a smoothed prior probability to maximum entropy model and used multiple features based on syntactic parsing. The smoothed prior is a combination of – through interpolation weight – a global distortion probability $p(o_k)$ and a local distortion probability $p(o_k|\bar{f}_n, \bar{e}_n)$ (i.e. lexicalised reordering model). The model predicts the jump distance (up to five words) from the previously translated source word to the current source word. This method does not capture the hierarchical structure of the sentence as explained by Galley and Manning (2008). The experiments were undertaken on a large-scale Chinese-English translation task (one million sentence pairs). The proposed model shows improvement over a distance-based reordering model. Like the findings of Zens and Ney (2006) and Xiong et al. (2006), there is no comparison with a lexicalised reordering model.

Ni et al. (2011) considered a variety of machine learning techniques including the maximum entropy model. They introduced a perceptron-based learning approach to modelling long-distance phrase movements. Similar to Xiang et al. (2011), their

model predicts the jump distance (up to five words) from the previously translated source word to the current one. Differing from the previous works, training data were divided into small independent sets where all samples share the same source phrase. This method breaks down the learning complexity by having as many sub-models as source phrases. Although the number of parameters for each sub-model are small, the total number of parameters are larger than having just one model to incorporate all the data. Several learning techniques are compared and evaluated on a Chinese-English corpus (Hong Kong laws corpus). The perceptron-based learning approach outperforms both the lexicalised reordering model and the maximum entropy model. The reported results were based on a test set taken from the same corpus.

Alrajeh and Niranjan (2014b) explored generative learning approach to phrase reordering in Arabic to English namely Bayesian naive Bayes. We achieved an improvement over a lexicalised reordering model. Training time of the model is as fast as the lexicalised one. Its storage requirement is many times smaller, which makes it more efficient particularly for large-scale tasks. Previously proposed discriminative models might achieve higher score than the reported results. However, the model is scalable since parameter estimation requires only one pass over the data with limited memory (i.e. no iterative learning). This is a critical advantage over discriminative models.

Recently, Cherry (2013) proposed using sparse features to optimise BLEU with the decoder instead of training a classifier independently. The reported results shows that sparse decoder features are superior to maximum entropy classifier.

We distinguish our work from the previous ones in the following. We propose fast and memory-efficient reordering models using multiclass SVM. In this study, we undertake a comparison between our work and both lexicalised and maximum entropy-based reordering models.

3. Baseline System

In statistical machine translation, the most likely translation $\boldsymbol{e}_{\text{best}}$ of an input sentence $\mathbf{f}$ can be found by maximising the probability $p(\boldsymbol{e}|\mathbf{f})$, as follows:

$$\boldsymbol{e}_{\text{best}} = \arg\max_{\boldsymbol{e}} p(\boldsymbol{e}|\mathbf{f}). \tag{1}$$

A log-linear combination of different models (features) is used for direct modelling of the posterior probability $p(\boldsymbol{e}|\mathbf{f})$ (Papineni et al., 1998; Och and Ney, 2002):

$$\boldsymbol{e}_{\text{best}} = \arg\max_{\boldsymbol{e}} \sum_{i=1}^{n} \lambda_i h_i(\mathbf{f}, \boldsymbol{e}), \tag{2}$$

where the feature $h_i(\mathbf{f}, \boldsymbol{e})$ is a score function over sentence pairs. The translation model and the language model are the main features in any system although additional features $h(.)$ can be integrated easily (such as word penalty).

In phrase-based systems, the translation model can capture the local meaning for each source phrase. However, to capture the whole meaning of a sentence, its translated phrases need to be in the correct order. The language model, which ensures fluent translation, plays an important role in reordering; however, the model is not sufficient (Al-Onaizan and Papineni, 2006). It prefers sentences that are grammatically correct without considering their actual meaning (i.e. the dependence of the target sentence on the source sentence). Besides that, it has a bias towards short translations[1] (Koehn, 2010). Therefore, developing a specific reordering model will improve the accuracy particularly when translating between two grammatically different languages.

4. Lexicalised Reordering Model

Phrase reordering modelling involves formulating phrase movements as a classification problem where each phrase position considered as a class (Tillmann, 2004). Some researchers classified phrase movements into three categories (monotone, swap, and discontinuous) but the classes can be extended to any arbitrary number (Koehn and Monz, 2005). In general, the distribution of phrase orientation is:

$$p(o_k|\bar{f}_i, \bar{e}_i) = \frac{1}{Z} h(\bar{f}_i, \bar{e}_i, o_k). \tag{3}$$

This lexicalised reordering model is estimated by relative frequency where each phrase pair $(\bar{f}_i, \bar{e}_i)$ with orientation o_k is counted and then normalised to yield the probability as follows:

$$p(o_k|\bar{f}_i, \bar{e}_i) = \frac{\text{count}(\bar{f}_i, \bar{e}_i, o_k)}{\sum_o \text{count}(\bar{f}_i, \bar{e}_i, o)}. \tag{4}$$

The orientation class of a current phrase pair is defined with respect to the previous target word or phrase (i.e. word-based classes or phrase-based classes). In the case of three categories (monotone, swap, and discontinuous): monotone is the previous source phrase (or word) that is previously adjacent to the current source phrase, swap is the previous source phrase (or word) that is next-adjacent to the current source phrase, and discontinuous is not monotone or swap. Galley and Manning (2008) extended the model to recognise sentence hierarchical structure.

5. Maximum Entropy-based Reordering Model

As mentioned in the introduction, maximum entropy classifier is a popular choice to model phrase movements. It is also known as multinomial logistic regression or

[1] In Moses, it is balanced by the word/phrase count features as noted by one of the reviewers.

softmax regression, which is a probabilistic model for the multiclass problem. The model is an extension of logistic regression which is a binary classifier. The class probability is given by:

$$p(o_k|\bar{f}_i, \bar{e}_i) = \frac{\exp(\boldsymbol{w}_k^\top \phi(\bar{f}_i, \bar{e}_i))}{\sum_{k'} \exp(\boldsymbol{w}_{k'}^\top \phi(\bar{f}_i, \bar{e}_i))}, \tag{5}$$

where $\phi(\bar{f}_i, \bar{e}_i)$ is a feature vector (see Table 3) and $\boldsymbol{w}_k$ is a weight vector measuring features' contribution to orientation o_k. The model's parameters are estimated by maximum likelihood. To do that, we write the function using the 1-of-K coding scheme in which $\mathbf{t}_i$ is a zero vector except where t_{ik} equals one, which indicates that an object is belonging to that class (Bishop, 2006). Then the likelihood is expressed as:

$$p(\mathbf{o}|\bar{\mathbf{f}}, \bar{\mathbf{e}}) = \prod_{i=1}^{N} \prod_{k=1}^{K} p(o_k|\bar{f}_i, \bar{e}_i)^{t_{ik}} \tag{6}$$

Now, taking the partial derivative of the log-likelihood we get (Bishop, 2006):

$$\frac{\partial \log L}{\partial \boldsymbol{w}_k} = \sum_{i=1}^{N} \left(t_{ik} - p(o_k|\bar{f}_i, \bar{e}_i)\right) \phi(\bar{f}_i, \bar{e}_i). \tag{7}$$

The solution is not closed-form but we can estimate $\boldsymbol{w}_k$ by the stochastic gradient descent. Similarly, MAP estimate can be used to impose regularisation on the parameters. In our experiments, we used a more advanced optimisation algorithm proposed by Andrew and Gao (2007)[2]. Their algorithm optimises L_1-regularised or L_2-regularised log-likelihood based on L-BFGS algorithm. The L_1 regularisation is equivalent to adding Laplacian prior over the model's parameters.

6. SVM-based Reordering Model

Phrase reordering problem is usually formulated as multiclass problem which can be solved as several binary problems in the standard SVM (Boser et al., 1992). One-versus-rest or one-versus-one are well known strategies.

In this work, we propose multiclass SVM to model phrase movements. We use dual coordinate method and a mechanism for pruning of the training samples, which allows us to train a reordering model efficiently. Before discussing our approach we briefly introduce multiclass SVM formulation.

Given a set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_i \in R^n$ and $y_i \in \{1, \ldots, K\}$, Crammer and Singer (2002) proposed a multiclass SVM formulation. Its dual optimisation problem is:

[2]We have used the authors' implementation of L-BFGS algorithm which is available at http://homes.cs.washington.edu/~galen/

$$\begin{aligned}
\underset{\boldsymbol{\alpha}}{\text{minimise}} \quad & \mathcal{D}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j=1}^{N}\alpha_{ik}\alpha_{jk}\boldsymbol{x}_i^T\boldsymbol{x}_j + \sum_{i=1}^{N}\sum_{k=1}^{K}(1-\delta_{ik})\alpha_i, \\
\text{subject to} \quad & \sum_{k=1}^{K}\alpha_{ik} = 0 \quad \text{and} \quad \alpha_{ik} \leq C\delta_{ik} \quad \forall i,k, \\
& \delta_{ik} = \begin{cases} 0 & \text{if } y_i \neq y_k; \\ 1 & \text{if } y_i = y_k. \end{cases}
\end{aligned} \tag{8}$$

where the corresponding $\boldsymbol{w}_k = \sum_{i=1}^{N}\alpha_{ik}\boldsymbol{x}_i$. Here $C \geq 0$ is a penalty parameter for margin violation by each data point $\boldsymbol{x}_i$.

For the sake of clarity, we use $\boldsymbol{x}_i$ to represent data in our discussion on SVM, and the learning algorithms. In the context of our NLP problem, and previous discussion in this paper $\boldsymbol{x}_i = \phi(\bar{f}_i, \bar{e}_i)$. Table 3 shows how a phrase pair can be represented.

Note that SVM is not a probabilistic classifier but in our experiments we used softmax function to yield a probabilistic decision (Bishop, 2006).

6.1. Shrinking dual method for solving Multiclass SVM

Keerthi et al. (2008) propose a sequential dual method to solve the problem (8). The method sequentially picks $\boldsymbol{x}_i$ at a time and optimises its dual variable (i.e. α_{ik} $\forall k$) while fixing all other variables. The sub-problem is given by:

$$\begin{aligned}
\underset{\boldsymbol{\alpha}_i}{\text{minimise}} \quad & \frac{1}{2}\sum_{k=1}^{K}\frac{1}{2}A\alpha_{ik}^2 + B_k\alpha_{ik}, \\
\text{subject to} \quad & \alpha_{ik} \leq C\delta_{ik} \quad \forall k,
\end{aligned} \tag{9}$$

where

$$\begin{aligned}
A = \boldsymbol{x}_i^T\boldsymbol{x}_i \quad \text{and} \quad B_k = G_{ik} - A\alpha_{ik}, \\
G_{ik} = \frac{\partial\mathcal{D}(\boldsymbol{\alpha})}{\partial\alpha_{ik}} = \boldsymbol{w}_k^T\boldsymbol{x}_i + 1 - \delta_{ik}.
\end{aligned} \tag{10}$$

Crammer and Singer (2002) provide $O(k \log k)$ algorithm to solve the sub-problem (9). Fan et al. (2008) present a simpler version given in Algorithm 1.

After each update, the corresponding weight vector for each class $\boldsymbol{w}_k$ is maintained as follows (Fan et al., 2008):

$$\boldsymbol{w}_k = \boldsymbol{w}_k + (\alpha_{ik}^{'} - \alpha_{ik})\boldsymbol{x}_i \tag{11}$$

Algorithm 1 Solving the sub-problem of multiclass SVM

Require: A, $\mathbf{B}$ and a penalty parameter $C \geq 0$

$D_k \leftarrow B_k + AC\delta_{ik}$, $\forall k$
Sort $\mathbf{D}$ in decreasing order
$\beta \leftarrow D_1 - AC$, $r \leftarrow 2$
while $r \leq K$ and $\beta/(r-1) < D_r$ **do**
    $\beta \leftarrow \beta + D_r$, $r \leftarrow r + 1$
end while
$\beta \leftarrow \beta/(r-1)$
$\alpha'_{ik} \leftarrow \min(C\delta_{ik}, (\beta - B_k)/A)$, $\forall k$

The optimal dual variables are achieved when the following condition is satisfied for all samples (Keerthi et al., 2008):

$$\nu_i = 0\,, \forall i, \quad \text{where} \quad \nu_i = \max_k G_{ik} - \min_{k:\alpha_{ik} < C\delta_{ik}} G_{ik}. \tag{12}$$

We propose a shrinking heuristic based on this condition which is a key to accelerate our algorithm. The dual variables α_{ik} are associated with each sample (i.e. phrase pair) therefore a training sample can be disregarded once its optimal dual variables are obtained. More data shrinking can be achieved by tolerating a small difference between the two values in (12). Algorithm 2 presents the overall procedure (shrinking step is from line 6 to 8).

Algorithm 2 Shrinking dual method for training large-scale multiclass SVM

Require: training set $S = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$

$\boldsymbol{\alpha} = 0$ and $\boldsymbol{w} = 0$
repeat
    Randomly pick i from S
    Calculate A_{ik} , B_{ik} , G_{ik} $\forall k$ by (10)
    Calculate ν_i by (12)
    if $\nu_i \leq \epsilon$ **then**
        Remove i from S
    else
        Calculate α'_{ik} $\forall k$ by Algorithm 1
        Update $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ by (11)
    end if
until $\nu_i \leq \epsilon$ $\forall i$

6.2. Kernel Mapping via Linear SVM

We have seen in the previous section that linear SVM can be scalable because of the advantage of accessing the feature space. On the other hand, kernel SVM is able to learn more complex patterns by working on high dimensional feature space, where the data might be linearly separable, without explicit mapping using the kernel trick.

An interesting technique to accelerate kernel SVM is to apply linear SVM to the explicit form. However, in many cases, kernel mapping is exponential to the input space or infinite as in the Gaussian kernel. Low-degree polynomial mapping is shown to be useful for certain datasets (Chang et al., 2010). All-subsets kernel is similar to polynomial kernel but has more flexibility in terms of the monomials' weightings (Shawe-Taylor and Cristianini, 2004). The mapping generates all combinations of input features and each monomial's coefficient equals one unlike polynomial mapping. Working with all monomials might be computationally expensive. Analysis of variance (ANOVA) kernel K_d, used in our experiments, restricts the mapping to subsets of cardinality d with $\binom{n}{d}$ dimensions (Shawe-Taylor and Cristianini, 2004). Table 3, in the next section, gives an example of AVOVA mapping.

7. Experiments

The Arabic-English parallel corpus used in our experiments is a combination of MultiUN, ISI and Ummah to set up a large-scale corpus. MultiUN is a large-scale parallel corpus extracted from the United Nations website[3] (Eisele and Chen, 2010). ISI and Ummah were taken from Linguistic Data Consortium[4] (LDC) with catalogue numbers (LDC2007T08) and (LDC2004T18), respectively. Table 1 shows general statistics of the corpora. Test sets are from NIST MT06 and MT08 where the Arabic sides are 1797 and 813 sentences, respectively. Each sentence has four English references.

Corpus	MultiUN		ISI		Ummah	
Statistics	Arabic	English	Arabic	English	Arabic	English
Sentence Pairs	9.7 M		1.1 M		80 K	
Running Words	255.5 M	285.7 M	30.5 M	34.4 M	2.7 M	2.9 M
Words/Line	22	25	27	31	33	36
Vocabulary Size	677 K	410 K	354 K	195 K	63 K	46 K
Vocabulary [%]	0.26	0.14	1.16	0.57	2.33	1.59

Table 1. General statistics of MultiUN, ISI and Ummah (M: million, K: thousand).

[3] http://www.ods.un.org/ods/

[4] http://ldc.upenn.edu/

We compare our approach with previously proposed reordering models in two phases. In the classification phase, we see the performance of the models as a classification problem. In the translation phase, we test the actual impact of these reordering models in a translation system.

7.1. Classification

We simplify the problem by classifying phrase movements into three categories (monotone, swap, discontinuous). To train the reordering models, we used GIZA++ to produce word alignments (Och and Ney, 2000). Then, we used the `extract` tool that comes with the Moses [5] toolkit (Koehn et al., 2007) in order to extract phrase pairs along with their orientation classes.

During the extraction process, each extracted phrase pair is represented by linguistic features. There are different feature representations in the literature as we have seen in Section 2. We explore a variety of feature sets as shown in Table 2. Each phrase pair is represented by all its words, its boundaries or its alignments. We have considered one or three words of context (i.e. occur before or after each phrase pair). Finally, one of ANOVA mappings were selected. Table 3 gives a generic example.

Feature	Phrase Pair			Context		ANOVA Mapping		
Set	all words	boundaries	alignments	size=1	size=3	d=1	d=2	d≤2
S1	●					✓		
S2		●				✓		
S3			●			✓		
S4		●		●		✓		
S5		●		●			✓	
S6		●		●				✓
S7			●	●		✓		
S8			●	●			✓	
S9			●	●				✓
S10		●			●	✓		
S11		●			●		✓	
S12		●			●			✓
S13			●		●	✓		
S14			●		●		✓	
S15			●		●			✓

Table 2. A variety of feature sets to represent a phrase pair.

[5] Moses is an open source toolkit for statistical machine translation (www.statmt.org/moses/).

Sentence pair:
Foreign sentence $\mathbf{f}$: $f_1\ f_2\ _1\ f_3\ f_4\ f_5\ _2\ f_6\ _3$.
English sentence $\mathbf{e}$: $e_1\ _1\ e_2\ e_3\ _3\ e_4\ e_5\ _2$.

Extracted phrase pairs $(\bar{f}, \bar{e})$:

$\bar{f}_i$	$\bar{e}_i$	o_i	alignments	
$f_1\ f_2$	e_1	mono	0-0	1-0
$f_3\ f_4\ f_5$	$e_4\ e_5$	swap	0-1	2-0
f_6	$e_2\ e_3$	other	0-0	0-1

Feature Representation:
a phrase pair is represented as a vector ϕ where each feature is a discrete number (0=not exist). Below is a representation of $\phi(\bar{f}_2, \bar{e}_2)$ in different feature sets:

S1 : f_3, f_4, f_5, e_4, e_5
S2 : f_3, f_5, e_4, e_5
S3 : f_3&e_5, f_5&e_4
S4 : f_3, f_5, e_4, e_5, f_2^-, f_6^+
S5 : $f_3\_f_5$, $f_3\_e_4$, $f_3\_e_5$, $f_3\_f_2^-$, $f_3\_f_6^+$, $f_5\_e_4$, $f_5\_e_5$, $f_5\_f_2^-$, $f_5\_f_6^+$, $e_4\_e_5$, $e_4\_f_2^-$, $e_4\_f_6^+$, $f_2^-\_f_6^+$
S6 : f_3, f_5, e_4, e_5, f_2^-, f_6^+, $f_3\_f_5$, $f_3\_e_4$, $f_3\_e_5$, $f_3\_f_2^-$, $f_3\_f_6^+$, $f_5\_e_4$, $f_5\_e_5$, $f_5\_f_2^-$, $f_5\_f_6^+$, $e_4\_e_5$, $e_4\_f_2^-$, $e_4\_f_6^+$, $f_2^-\_f_6^+$

Table 3. A generic example of the process of phrase pair extraction and representation in different feature sets

Firstly, we present the performance of lexicalised reordering model in Table 4. Then, we compare MaxEnt and multiclass SVM under all feature sets in Table 2. It is not hard to see that using MaxEnt with an alternate feature set that enumerates all conjunctions of size d is equal to ANOVA mapping. Tables 5 and 6 report the results.

Orientation	Confusion Matrix			Accuracy	Precision	Recall	F_1 score
	Mono	Swap	Disc.	all classes			
Monotone	68.9	0.9	1.3		97.0	77.0	85.9
Swap	6.4	2.6	0.8	75.9	26.8	63.5	37.7
Discontinuous	14.2	0.6	4.4		23.0	68.4	34.5

Table 4. The performance of lexicalised reordering model.

Feature Set	Time	Acc.	Precision			Recall			F_1 score		
			M	S	D	M	S	D	M	S	D
S1	1h26m	74.1	96.6	17.3	19.7	75.5	57.7	61.4	84.8	26.6	29.8
S2	1h10m	74.0	**97.1**	13.9	19.1	75.1	**61.2**	**62.0**	84.7	22.7	29.2
S3	1h40m	**76.1**	93.5	**32.1**	**34.0**	**79.8**	55.5	58.6	**86.1**	**40.7**	**43.0**
S4	1h50m	77.0	**95.5**	37.3	29.1	78.9	69.6	63.0	86.4	48.5	39.9
S5	5h59m	80.7	94.9	49.2	44.2	83.3	71.9	**68.4**	88.7	58.4	53.7
S6	6h21m	**81.3**	94.4	**53.3**	**46.9**	**84.3**	**72.5**	67.6	**89.1**	**61.4**	**55.4**
S7	3h10m	78.7	93.8	45.2	39.7	82.2	67.1	61.7	87.6	54.0	48.3
S8	4h32m	81.4	**93.9**	51.6	50.7	85.0	**72.7**	66.7	89.3	60.4	57.6
S9	4h43m	**82.5**	93.4	**59.5**	**53.9**	**86.7**	72.1	**67.3**	**89.9**	**65.2**	**59.9**
S10	2h45m	76.2	94.1	34.0	31.3	79.2	61.9	58.8	86.0	43.9	40.9
S11	15h11m	82.4	**95.2**	56.4	47.3	**84.8**	**75.0**	**74.0**	**89.7**	64.4	57.7
S12	16h04m	**82.6**	94.9	**58.1**	**48.9**	84.7	73.3	71.2	89.5	**64.8**	**58.0**
S13	3h24m	78.8	92.3	46.3	45.1	83.8	62.2	59.8	87.9	53.1	51.4
S14	13h03m	82.2	**93.9**	50.0	45.4	83.5	**78.6**	68.8	88.4	61.1	54.7
S15	15h12m	**82.9**	93.4	**59.8**	**54.8**	**88.3**	72.8	**69.9**	**90.8**	**65.7**	**61.4**

Table 5. Maximum entropy-based reordering model's performance (M is monotone, S is swap, D is discontinuous). The reported time is in hours (h) and minutes (m).

Feature Set	Time	Acc.	Precision			Recall			F_1 score		
			M	S	D	M	S	D	M	S	D
S1	0h30m	70.8	92.7	7.4	22.3	76.5	31.9	36.8	83.8	12.0	27.8
S2	0h28m	71.7	**96.2**	13.3	10.7	74.9	30.9	44.5	84.2	18.6	17.3
S3	0h40m	**75.8**	93.3	**36.3**	**31.1**	**80.0**	**50.5**	**58.8**	**86.1**	**42.2**	**40.7**
S4	0h33m	75.6	**95.9**	35.9	21.0	77.7	59.9	62.2	85.8	44.9	31.4
S5	1h45m	82.1	95.8	55.8	44.8	84.1	**73.7**	**73.3**	89.6	63.5	55.6
S6	2h07m	**82.5**	95.1	**60.0**	**47.4**	**85.2**	72.1	72.4	**89.9**	**65.5**	**57.3**
S7	0h47m	79.3	93.7	49.5	41.3	82.8	69.0	62 .7	87.9	57.7	49.8
S8	1h24m	81.0	**95.3**	50.4	42.9	83.3	69.0	**71.0**	88.9	58.2	53.5
S9	1h41m	**82.1**	92.5	**61.0**	**54.1**	**86.8**	**69.7**	65.6	**89.6**	**65.1**	**59.3**
S10	0h44m	74.0	95.6	24.5	18.7	76.8	49.2	53.4	85.2	32.7	27.7
S11	4h33m	**82.7**	**95.9**	56.2	47.4	84.7	**75.0**	**74.2**	89.8	64.3	57.9
S12	4h51m	82.6	94.9	**57.9**	**49.7**	**85.4**	73.3	71.8	**89.9**	**64.7**	**58.7**
S13	0h59m	78.0	94.2	46.3	35.0	81.6	58.4	62.2	87.4	49.7	44.8
S14	3h32m	82.0	**96.8**	49.1	44.0	83.2	**77.3**	**75.8**	89.5	60.0	55.6
S15	4h04m	**82.8**	95.5	**55.8**	**49.8**	**85.4**	73.6	72.8	**90.2**	**63.5**	**59.1**

Table 6. Multiclass SVM-based reordering model's performance.

Four observations can be drawn from the results in Table 5 and Table 6. First, the performance of multiclass SVM is similar to MaxEnt in most feature sets. Second, our classifier is a couple of times faster than MaxEnt (around 4-fold) due to the shrinking method. Third, context around phrase pairs is important to achieve high accuracy and only one word before and after is enough. Finally, alignment features usually have higher F_1 score than boundary features in both MaxEnt and multiclass SVM.

Alrajeh and Niranjan (2014a) propose a dual multinomial logistic regression (Dual MLR) with a shrinking heuristic to model phrase movements. We compare their shrinking approach with multiclass SVM in Figure 1.

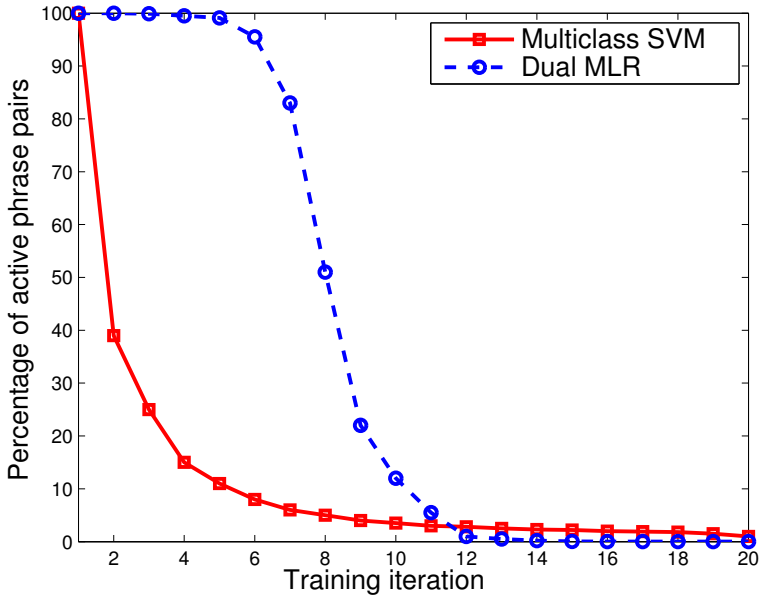


Figure 1. Comparison between multiclass SVM and dual multinomial logistic regression (MLR) in terms of active phrase pairs during training.

In Figure 2, we show training time and memory usage for each classifier (Multiclass SVM, Dual MLR, MaxEnt) when the number of phrase pairs increases. The results show that multiclass SVM consumes mush less memory (nearly half) than MaxEnt due to the shrinking technique discussed in Section 6.1.

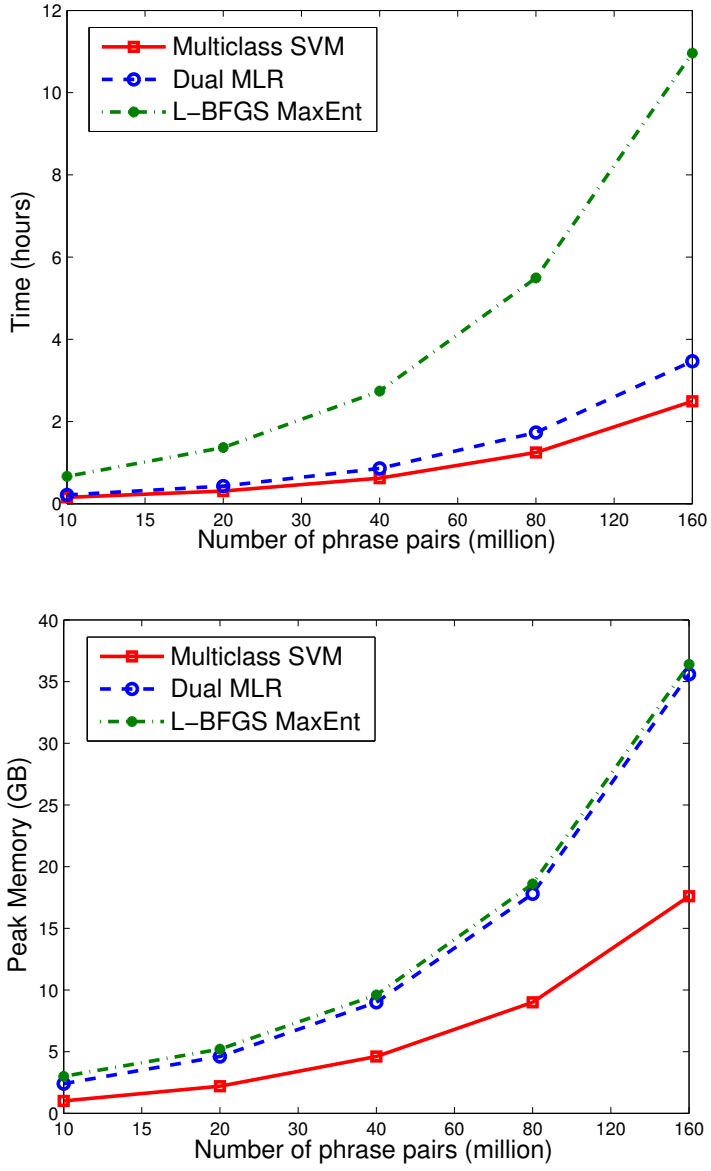


Figure 2. Training time (above) and memory usage (below) for each classifier when the number of phrase pairs increases.

7.2. Translation

We used the Moses toolkit (Koehn et al., 2007) with its default settings. The language model is a 5-gram built from the English side with interpolation and Kneser-Ney smoothing (Kneser and Ney, 1995). We tuned the system using PRO technique (Hopkins and May, 2011). We built seven Arabic-English translation systems. The first system has no reordering model, only a distortion penalty. The second system has a hierarchical lexicalised reordering model that is built by specifying the configuration string `hier-msd-backward-fe`. Sparse reordering features (Cherry, 2013) are added in the third system. We only used '`sparse-phrase=1`' option with top 200 words. The last four systems have SVM-based or MaxEnt-based reordering models.

As commonly used in statistical machine translation, we evaluated the translation performance by BLEU score (Papineni et al., 2002) and NIST (Doddington, 2002). We also computed statistical significance for the proposed models using a *paired bootstrap resampling* method (Koehn, 2004b).

Table 7 reports the size of each reordering model. Note that there is a big difference between the lexicalised model and the discriminative ones.

Reordering Model	Lexicalised	Multiclass SVM (S6)	Multiclass SVM (S7)
Parameters (million)	73.2	17.1	2.4
Disk Storage (GB)	5.9	0.7	0.1

Table 7. Comparison of problem sizes for the different reordering models.

Table 8 presents NIST and BLEU scores for five translation systems in MT06 and MT08 test sets. Our models achieve improvements on top of a strong baseline system with sparse reordering features. Note that feature sets (S6) and (S7) have similar scores although (S6) has higher classification accuracy in Table 6.

	MT06				MT08			
Phrase-based SMT	NIST	Δ	BLEU	Δ	NIST	Δ	BLEU	Δ
No Reordering Model	9.1	-0.3	35.5	-1.6	9.9	-0.2	41.2	-1.7
LexicalRM (**baseline**)	9.4	–	37.1	–	10.1	–	42.9	–
LexicalRM + sparseRM	9.5	+0.1	37.6	+0.5	10.2	+0.2	43.8	+0.9
SVM-RM (S6) + sparseRM	9.6	+0.2	38.1	+1.0	10.4	+0.3	44.4	+1.5
SVM-RM (S7) + sparseRM	9.6	+0.2	38.0	+0.9	10.4	+0.3	44.3	+1.4
MaxEnt-RM (S6) + sparseRM	9.6	+0.2	38.1	+1.0	10.4	+0.3	44.4	+1.5
MaxEnt-RM (S7) + sparseRM	9.6	+0.2	38.1	+1.0	10.4	+0.3	44.4	+1.5

Table 8. NIST and BLEU [%] scores for two evaluation sets (RM: Reordering Model).

8. Conclusion

Posing phrase movements as a classification problem, we exploit recent developments in solving large-scale multiclass support vector machines using stochastic gradient learning algorithm and show significant advantages in Arabic-English systems. The algorithms we propose are shown to be computationally fast and memory-efficient. In terms of evaluating translation quality using the BLEU score, we achieve 1.0 point in MT06 and 1.5 in MT08 over a lexicalised reordering model with at least 95% statistical significance. Our SVM-based model is shown to be superior to the maximum entropy-based model. It is a couple of times faster (nearly 4-fold) and more memory-efficient (50% reduction).

The expanded space due to ANOVA mapping can be reduced significantly by removing less frequent features. We found that a reordering model based on alignments features (S7) is more compact than using boundaries features (S6).

Our current work focuses on two issues. The first issue is exploring higher degrees of ANOVA kernels and others in order to reduce the classification error rate. The second issue is reducing feature space by using limited but informative features such as part-of-speech tags.

Acknowledgements

The first author was funded by a scholarship from King Abdulaziz City for Science and Technology (KACST).

Bibliography

Al-Onaizan, Yaser and Kishore Papineni. Distortion models for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 529–536, Sydney, Australia, July 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P06/P06-1067.

Alrajeh, Abdullah and Mahesan Niranjan. Large-scale reordering model for statistical machine translation using dual multinomial logistic regression. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1758–1763. Association for Computational Linguistics, 2014a. URL http://aclweb.org/anthology/D14-1183.

Alrajeh, Abdullah and Mahesan Niranjan. Bayesian reordering model with feature selection. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 477–485, Baltimore, Maryland, USA, June 2014b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3361.

Alrajeh, Abdullah, Akiko Takeda, and Mahesan Niranjan. Memory-efficient large-scale linear support vector machine. In *Proceedings of SPIE: Seventh International Conference on Machine Vision (ICMV 2014)*, volume 9445, pages 944527–944527–6, Milan, Italy, February 2015. SPIE. doi: 10.1117/12.2180925. URL http://dx.doi.org/10.1117/12.2180925.

Andrew, Galen and Jianfeng Gao. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 33–40. ACM, 2007. ISBN 978-1-59593-793-3.

Bishop, Christopher M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, 1992.

Brown, P., J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. A statistical approach to language translation. In *12th International Conference on Computational Linguistics (COLING)*, pages 71–76, 1988.

Brown, Peter F., John Cocke, Stephen A. Della-Pietra, Vincent J. Della-Pietra, Frederick Jelinek, Robert L. Mercer, and Paul Rossin. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Chang, Yin-Wen, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 2:1471–1490, Apr. 2010.

Cherry, Colin. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N13-1003`.

Crammer, Koby and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, Mar. 2002.

Cristianini, Nello and John Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000. ISBN 0-521-78019-5.

Doddington, George. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

Eisele, A. and Y. Chen. MultiUN: A multilingual corpus from united nation documents. In Tapias, Daniel, Mike Rosner, Stelios Piperidis, Jan Odjik, Joseph Mariani, Bente Maegaard, Khalid Choukri, and Nicoletta Calzolari (Conference Chair), editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation*, pages 2868–2872. European Language Resources Association (ELRA), 5 2010.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Galley, Michel and Christopher D. Manning. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Hawaii, October 2008. Association for Computational Linguistics.

Glasmachers, Tobias and Ürün Dogan. Accelerated coordinate descent with adaptive coordinate frequencies. In Ong, Cheng Soon and Tu Bao Ho, editors, *Asian Conference on Machine Learning, ACML*, volume 29 of *JMLR Proceedings*, pages 72–86. JMLR.org, 2013.

Hopkins, Mark and Jonathan May. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D11-1125`.

Hsieh, Cho-Jui, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 408–415, 2008.

Keerthi, S. Sathiya, Sellamanickam Sundararajan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A sequential dual method for large scale multi-class linear SVMs. In *Proceedings of the Forteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 408–416, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/sdm_kdd.pdf`.

Kneser, Reinhard and Hermann Ney. Improved backing-off for m-gram language modeling. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 181–184, 1995.

Koehn, Philipp. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of 6th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 115—124, Washington DC, 2004a.

Koehn, Philipp. Statistical significance tests for machine translation evaluation. In Lin, Dekang and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July 2004b. Association for Computational Linguistics.

Koehn, Philipp. *Statistical Machine Translation*. Cambridge University Press, 2010.

Koehn, Philipp and Christof Monz. Shared task: Statistical machine translation between European languages. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*, pages 119–124. Association for Computational Linguistics, 2005.

Koehn, Philipp, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of International Workshop on Spoken Language Translation*, Pittsburgh, PA, 2005.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Christopher J. Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180, 2007.

Kumar, Shankar and William Byrne. Local phrase reordering models for statistical machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 161–168, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

Nguyen, Vinh Van, Akira Shimazu, Minh Le Nguyen, and Thai Phuong Nguyen. Improving a lexicalized hierarchical reordering model using maximum entropy. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation, 2009.

Ni, Y., C. Saunders, S. Szedmak, and M. Niranjan. Exploitation of machine learning techniques in modelling phrase movements for machine translation. *Journal of Machine Learning Research*, 12:1–30, Feb. 2011. ISSN 1532-4435.

Och, Franz Josef and Hermann Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association of Computational Linguistics (ACL)*, 2000.

Och, Franz Josef and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2002.

Och, Franz Josef and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

Papineni, K.A., S. Roukos, and R.T. Ward. Maximum likelihood and discriminative training of direct translation models. In *Proceedings of ICASSP*, pages 189–192, 1998.

Papineni, K., S. Roukos, T. Ward, and W. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Stroudsburg, PA, USA, 2002. ACL.

Shawe-Taylor, John and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

Tillmann, Christoph. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL: Short Papers*, pages 101–104, 2004.

Xiang, Bing, Niyu Ge, and Abraham Ittycheriah. Improving reordering for statistical machine translation with smoothed priors and syntactic features. In *Proceedings of SSST-5, Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 61–69, Portland, Oregon, USA, 2011. Association for Computational Linguistics.

Xiong, Deyi, Qun Liu, and Shouxun Lin. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 521–528, Sydney, July 2006. Association for Computational Linguistics.

Zens, Richard and Hermann Ney. Discriminative reordering models for statistical machine translation. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 55–63, New York City, June 2006. Association for Computational Linguistics.

Address for correspondence:
Abdullah Alrajeh
asar1a10@ecs.soton.ac.uk
School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom, SO17 1BJ

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015 85–110

Evaluating Machine Translation Quality Using Short Segments Annotations

Matouš Macháček, Ondřej Bojar

Charles University in Prague, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics

Abstract

We propose a manual evaluation method for machine translation (MT), in which annotators rank only translations of short segments instead of whole sentences. This results in an easier and more efficient annotation. We have conducted an annotation experiment and evaluated a set of MT systems using this method. The obtained results are very close to the official WMT14 evaluation results. We also use the collected database of annotations to automatically evaluate new, unseen systems and to tune parameters of a statistical machine translation system. The evaluation of unseen systems, however, does not work and we analyze the reasons.

1. Introduction

Manual evaluation is considered as the only source of truth, which automatic metrics try to approximate. However, it suffers from many disadvantages. Since it includes manual labour, it is very costly and slow. Moreover, manual evaluation is not reproducible with exactly the same results; human annotators have different criteria for comparing candidates and even an individual annotator is not consistent with himself or herself in time. Human evaluation is therefore most often used in shared evaluation campaigns and sometimes used by MT developers when a new component of a system needs to be evaluated. It is definitely not feasible to directly use human evaluation in an automatic method for tuning a model's parameters because these methods require to evaluate millions of sentences.

It would be very useful to have an evaluation method with the advantages of both manual and automatic methods. Recently, there actually emerged methods on the

 Corresponding author: bojar@ufal.mff.cuni.cz
Cite as: Matouš Macháček, Ondřej Bojar. Evaluating Machine Translation Quality Using Short Segments Annotations. The Prague Bulletin of Mathematical Linguistics No. 103, 2015, pp. 85–110.
doi: 10.1515/pralin-2015-0005.

boundary of manual and automatic evaluation. See Zaidan and Callison-Burch (2009) and Bojar et al. (2013) for example of such methods. These methods usually require a large manual annotation effort at the beginning to create a database of annotations and then they use the collected database in an automatic way during evaluation of unseen sentences.

The goal of this article is to propose a method, which could be used to manually evaluate a set of systems and the database collected during this manual evaluation could be later reused to automatically evaluate new, unseen systems and to tune the parameters of MT systems. This goal includes, besides proposing the method, also developing an annotation application, conducting a real evaluation experiment and experimenting with reusing the collected database.

The article is organized as follows. In Section 2, we propose the manual evaluation method. In Section 3, we describe our annotation experiment with the proposed method and evaluate annotated systems. We explore the possibility of reusing the collected database to evaluate new systems in Section 4. And finally, we try to employ the collected database in tuning of an MT system in Section 5.

2. SegRanks: Manual Evaluation Based on Short Segments

In the WMT official human evaluation (Bojar et al., 2014),[1] humans annotate whole sentences. They are presented with five candidate translations of a given source sentence and their task is to rank these candidates relative to one another (ties are allowed). One of the disadvantages of this method is that the sentences are quite long and therefore quite hard to remember for the annotators to compare them. Also, when comparing longer sentences, there are many more aspects, in which one sentence can be better or worse than another, and therefore it is more difficult for the annotators to choose the better of the candidates.

To avoid these disadvantages, we propose the following method. Instead of judging whole sentences, we extract short segments[2] from candidates and present them to annotators to rank them. In order to extract meaningful segments with the same meaning from all candidates, we do the following procedure: First, we parse the source sentence and then recursively descend the parsed tree and find nodes that cover source segments of a given maximum length. This is captured in Algorithm 1. Finally, we project these extracted source segments to their counterpart segments in all candidate sentences using an automatic alignment.[3] The whole process is illustrated in Figure 1. This extraction method is inspired by Zaidan and Callison-Burch

[1] `http://www.statmt.org/wmt15` and previous years

[2] The term 'segment' is sometimes used in the literature to refer a sentence. In this article, we will use the term 'segment' for a phrase of few words.

[3] We allow gaps in the projected target segments.

(2009) and by the constituent ranking technique as it was used in WMT07 manual evaluation (Callison-Burch et al., 2007).

Algorithm 1 Short segment extraction from source-side parse tree

```
1: function ExtractSegments(treeNode, minLength, maxLength)
2:     extractedSegments ← list()
3:     leaves ← treeNode.leaves()
4:     if length(leaves) ≤ maxLength then
5:         if lenth(leaves) ≥ minLength then
6:             extractedSegments.append(leaves)
7:     else
8:         for node in treeNode.children() do
9:             segments ← ExtractSegments(child, minLength, maxLength)
10:            extractedSegments.extend(segments)
       return extractedSegments
```

In the constituent ranking in WMT07, these extracted segments were only highlighted and shown to the annotators together with the rest of the sentence. The annotators were asked to rank the highlighted segments in the context of the whole candidate sentences.

We use a different approach here, which is more similar to that used by Zaidan and Callison-Burch (2009). We show the extracted segments without any context and ask the annotators to rank them. The only additional information provided to the annotators is the whole source sentence with the source segment highlighted. The annotators are told that they can imagine any possible translation of the source sentence where the ranked segment fits best. They are instructed to penalize only those segments for which they cannot imagine any appropriate rest of the sentence.

While we are aware that this approach has some disadvantages (which we summarize below), there is one significant advantage: it is much more likely that two systems produce the same translation of a short segment than that they would produce the same translation of a whole sentence. Because we do not show the sentence context, we can merge identical segment candidates into one item, so the annotators have fewer candidate segments to rank. This also allows us to reuse already collected human annotations later to evaluate a new system that was not in the set of annotated systems or to tune parameters of a system.

The following list summarizes known disadvantages of this method. However, we believe that the advantages still outweigh the problems and that our method is worth exploration.

- A system can translate shorter segments quite well but it can fail to combine them properly when creating the whole sentence translation. For instance, a system may fail to preserve the subject-verb agreement. In their paper, Zaidan

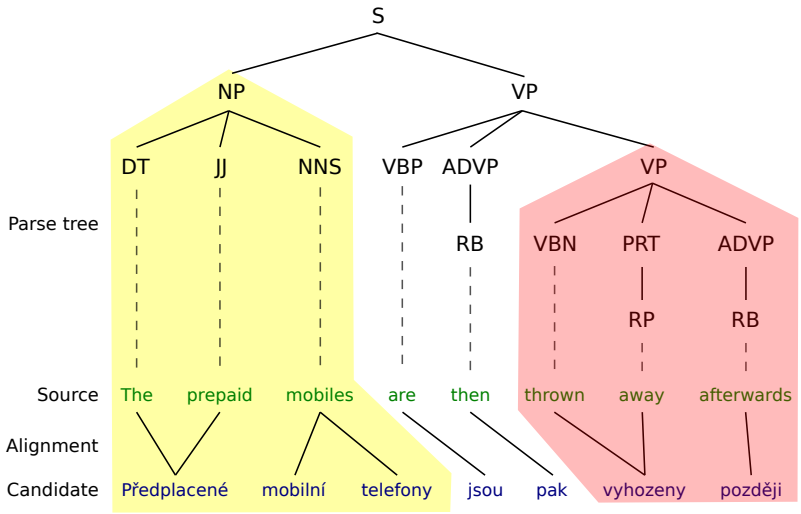


Figure 1: An illustration of candidate segments extraction process. For a given MT system, two segments were extracted from this sentence. The `maxLength` constant was set to the value 4 here, to illustrate that not all of the words are always covered by the extracted segments.

and Callison-Burch (2009) suggest to go up the parse tree and extract also the longer segments that consist of already extracted shorter segments. However, if we use this approach the amount of annotation work would multiply several times. Furthermore, the longer segments are more difficult to rank and the chance that systems' candidates will be identical (so that we can merge them for annotation) is lower.

- The annotators do not see the whole context of annotated short segments. They are instructed to imagine any suitable context of the segment. However, they can fail to imagine a suitable context even if there exists one and wrongly penalize the segment. To partially remedy this disadvantage, we give all extracted short segments to the annotators to judge at once as a source of inspiration.
- Extracted short segments do not cover the whole sentence. For example in Figure 1, the words 'jsou' and 'pak' are not part of any extracted segment. We would avoid this problem if we set the variable `minLength` to zero. This, however, would generate a high number of short segments to annotate.
- Some segments are much more important in conveying the meaning of a sentence than others, and therefore they should not have equal weights when they are considered in scoring. When an annotator ranks system A better than system B in two of three ranked segments, and system B better than system A in

the third segment, it does not always mean that he would have ranked system A better than system B when ranking the whole sentences. The third segment could be much more important for the quality of translation than the first two. We are afraid that it is not possible to easily avoid this problem. However, we also believe that this problem is not so severe and that possible differences in the importance of individual segments cancel out.

- The word-alignments are computed automatically and are not always correct. The projected target segments may not exactly correspond to the source segments and may mislead the annotators.

3. Experiments

3.1. Data and Segment Preparation

We used English to Czech part of the WMT14 (Bojar et al., 2014) test set. We chose this dataset to be able to compare our results with the official WMT14 human evaluation.

The test set consists of 3 003 sentences (68 866 English tokens). It contains both source sentences and reference translations. Besides that, we also used candidate translations of all 10 systems that participated in the English → Czech WMT14 translation task.

The source sentences and all candidate translations were tokenized and normalized. The source sentences were parsed using the Stanford parser. We used lexicalized `englishFactored` model (Klein and Manning, 2003b), which is distributed with the parser. We also tried unlexicalized `englishPCFG` (Klein and Manning, 2003a) and compared the segments extracted using the both parsers on a small random sample of sentences. The `englishFactored` model yielded subjectively better segments.

We constructed the word alignment between the source sentences and the candidate translations using Giza++ (Och and Ney, 2003). Since the alignment algorithm is unsupervised and the amount of all candidate translations is relatively small ($10 \times 3\,003$), we introduced more data by concatenating all candidate translations with 646 605 sentences taken from the Europarl parallel corpus (Koehn, 2005) and with 197 053 sentences taken from the CzEng parallel corpus (Bojar et al., 2012). We used grow-diag-final symetrization method (Koehn et al., 2003) to combine alignments computed in both directions.

We extracted short segments from the parsed source trees using Algorithm 1. The constant `minLength` was set to the value 3 to filter out very short segments, most of which are hard to compare without context. This also helped reduce the number of extracted segments to be annotated. The constant `maxLength` was set to the value 6 so the extracted segments were not too long to compare and at the same time it was more likely that two candidate translations of a segment were equal and thus there would be fewer items to rank (our aim was to make annotations as easy and fast as

possible). We have experimented with various settings of these two constants and the final setting seemed to generate a reasonable number of meaningful segments.

From the 3 003 source sentences, we have extracted 8 485 segments of length 3 to 6 tokens. That is approximately 2.83 segments per sentence on average. The extracted segments covered 54.9 % of source tokens. By projecting the source segments to the candidate sentences using the computed alignments, we obtained $10 \times 8\,485 = 84\,850$ candidate segments. However, after the merging of identical segments, only 50 011 candidate segments were left. This represents 58.9 % of the original candidate segments, or in other words, after the merging we got 5.89 (instead of original 10) candidate segments to be ranked for each source segment on average. These candidate segments were inserted into the database to be ranked by the annotators.

3.2. Ranking of Segments

We have developed a new annotation application called *SegRanks* for this annotation experiment. An example screenshot is given in Figure 2. Annotation instructions were displayed on each annotation screen. This is the English translation of these instructions:

> A number of segments are extracted from the annotated sentence. You are shown a few candidate translations for each of these segments. Your task is to distinguish acceptable candidate translations (the meaning of the segment can be guessed despite a few or more errors) from unacceptable ones (the meaning is definitely not possible to guess from the candidate segment). Also please rank the acceptable candidate translations relatively from the best ones to the worst ones. Please place better candidate translations higher and the worse ones lower. You can place candidates of the same quality on the same rank. We ask that you place unacceptable candidates to the position "Garbage".
>
> Please note that source segments and their candidate translations are chosen automatically and do not have to be perfect. Consider them only as approximate clues. If a candidate segment contains an extra word that does not correspond to the source segment but otherwise could be in the translated sentence, you do not have to rank such candidate any worse. If something is missing in the candidate translation you should consider it an error.

Our goal was to make the annotation as efficient and user friendly as possible. The annotators rank all the segments of a sentence on a single screen (so that they have to read the whole source sentence and reference translation only once). For each annotated segment, they see the source sentence repeated, with the annotated segment highlighted. The annotators rank the segment candidates by dragging and dropping them to appropriate rank positions. When all the candidates of all the source segments of the sentence are ranked, the annotators submit their annotations to the

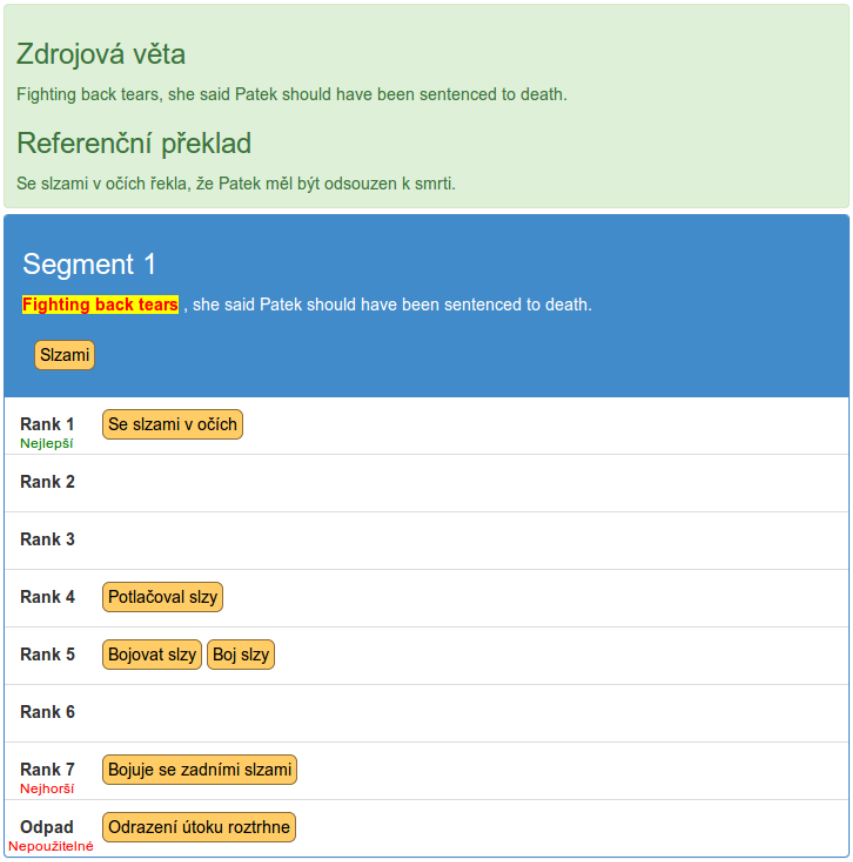


Figure 2: A screenshot of the annotation interface in Czech. The annotators rank the candidate segments by dragging and dropping them into the ranks. The annotators see all annotated segments of a sentence on a single screen.

server. The web interface has a responsive design, so it is displayed correctly on smaller screens, and the drag-and-drop works also on touch screens. The annotators were therefore able to rank segments on a tablet.

During the annotation experiment, 17 annotators ranked segments of 2 765 sentences, which is more than 92 % of the prepared English-Czech test set.

3.3. Annotator Agreements

To measure the reliability and robustness of the proposed annotation method, we compute intra- and inter-annotator agreements. A reasonable degree of these agreements supports the suitability of this method for machine translation evaluation.

We measured the agreements using Cohen's kappa coefficient (κ) (Cohen, 1960). Let $P(A)$ be the proportion of times the annotators agree and $P(E)$ be the proportion of time that they would agree by chance. Then the Cohen's κ is computed using the following formula:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

In our case, $P(A)$ and $P(E)$ are computed in the context of pairwise comparisons. Approximately 5 % of the annotated sentences were annotated twice by two different annotators (for the inter-annotator agreement). Another 5 % of the sentences were annotated twice by the same annotator (for the intra-annotator agreement). From all the segments of these double annotated sentences, we extracted pairwise comparisons of candidate segments. Then we computed $P(A)$ as the proportion of pairwise comparisons in which annotations match.

We computed the expected agreement by chance as

$$P(E) = P(A > B)^2 + P(A = B)^2 + P(A < B)^2$$

where $P(A = B)$ was computed empirically as the relative frequency of cases where two segments were ranked equaly (across all annotations of all segments of all sentences, regardless the annotator) and the other two were computed as $P(A < B) = P(A > B) = \frac{1-P(A=B)}{2}$. The value of $P(E)$ in our experiment is 0.394, which means that the probability of the outcomes $A > B$, $A = B$ and $A < B$ is not uniform.

The final values of inter-annotator and intra-annotator κ can be found in Table 1. For comparison, we report the corresponding κ values from WMT14 sentence ranking task (Bojar et al., 2014), which were computed identically on the same test set. The exact interpretation of the Kappa coefficient is difficult, but according to Landis and Koch (1977), values in the range 0–0.2 indicate a slight agreement, 0.2–0.4 fair, 0.4–0.6 moderate, 0.6–0.8 substantial and 0.8–1.0 almost perfect agreement. Our method obtains both κ scores better than full sentence ranking in WMT14. However, they are still quite low given that our annotation was designed to be much simpler than the ranking full sentences.

	our method	Bojar et al. (2014)
intra-annotator κ	0.593	0.448
inter-annotator κ	0.397	0.360

Table 1: κ scores measuring intra-annotator and inter-annotator agreements. We also report corresponding κ scores from official WMT translation task for comparison.

3.4. Overall Ranking of Annotated Systems

In the first experiment, we would like to show that the proposed method can be used to produce overall ranking of the annotated systems, which are very similar to the official human evaluation in WMT.

To compute an overall score for each system, we use the method **Ratio of wins (ignoring ties)** from WMT12 (Callison-Burch et al., 2012). We interpret segment ranks as pairwise comparisons: for each short segment candidate we compute how many times it was ranked better than another segment and how many times it was ranked worse. For a given candidate translation C, we can sum up these counts over all short segments of all the sentences to get total number of pairwise wins $win(C)$ and total number of pairwise losses $loss(C)$. The overall score is then computed using this formula:

$$E_{win}(C) = \frac{win(C)}{win(C) + loss(C)}$$

Table 2a reports the overall scores based on the short segment annotations. To compare our method with the classical method of annotating whole sentences, we apply the same method to the annotations collected during WMT14 manual evaluation, see Table 2b.

The overall rankings of the systems obtained from both types of annotation are very similar. However, there are two changes when comparing to the sentence-level results: the system online-B is better and system cu-bojar is worse according to the segments-level results.

3.5. Analysis

For the explanation of the differences between the overall rankings computed on sentence-level and segment-level annotation, we have to look into the data. We extract candidate translations that were ranked high by segment-level annotation but ranked low by the sentence-level annotation. In the following, we present some of these sentences with comments. The ranked segments are in bold.

System	Score
cu-depfix	0.5777
onlineB	0.5642
uedin-unconstrained	0.5626
cu-bojar	0.5606
cu-funky	0.5566
uedin-wmt14	0.5498
onlineA	0.5007
CU-TectoMT	0.4485
commercial1	0.3992
commercial2	0.3492

(a) Short segments annotation

System	Score
cu-depfix	0.6101
cu-bojar	0.6011
uedin-unconstrained	0.5967
cu-funky	0.5823
onlineB	0.5439
uedin-wmt14	0.5285
onlineA	0.5039
CU-TectoMT	0.4473
commercial1	0.3617
commercial2	0.2780

(b) Official WMT14 annotation

Table 2: Overall rankings of systems according to **Ratio of wins (ignoring ties)** score based on (a) our short segment annotations and (b) the official WMT14 full sentence annotations for comparison.

Source: Airlines began charging for **the first and second checked bags** in 2008.
Candidate: Letecké linky začaly nabíjení pro **první a druhý odbavených zavazadel** v roce 2008.
(Sentence 715, online-B)

The translation of the compared segment is relatively good, the case of the noun phrase is wrong but the meaning could be understood. The reason why the whole sentence was ranked poorly is probably the word "nabíjení" ("Charging a battery" in English), which is obviously a wrong lexical choice of the MT system. Unfortunately, this word is not covered by the only ranked segment. A similar problem is also in the following sentence:

Source: I want to fulfil **my role of dad and husband**.
Candidate: Chci, aby splnil **svou roli táty a manžela**.
(Sentence 559, cu-bojar)

The translation of the segment is perfect but the subject of the candidate translation is wrongly expressed as the third person. The whole sentence is therefore correctly ranked as a poor translation. And again, this is not covered by the evaluated segment.

In the two previous examples, the predicate was wrongly translated but unfortunately, it was not covered in the extracted segments and therefore reflected in segment-level annotation. This seems to be the main disadvantage of our method: extracted

segments sometimes do not cover predicates, which are very important for the annotator when judging the overall quality of the sentence.

4. Evaluating New Systems

Machine translation systems often translate a given short source segment identically. This was one of our main motivations for ranking translations of short segments. As we showed in the previous section, evaluating the annotated systems using short segments annotations works reasonably well (despite the shortcomings described above). It would be very useful if we could reuse the database of annotations to evaluate also unseen systems. The more annotated systems we have in the database the more likely it is that an unseen MT system produces a translation of a short segment that is already in the database. We will call this situation a **hit**. If the translated segment is not yet annotated, we will call it a **miss**.

Because we don't have any spare systems that were not annotated, we use a variant of cross validation, called **leave-one-out**: in each step, we choose one system and remove its segments from the database of annotations. Then we treat this system as an unseen one and try to match the system's segments with the segments in the database.

4.1. Exact Matching of Candidate Segments

The most obvious way to evaluate an unseen translation is to compute the **Ratio of wins (ignoring ties)** of all the systems (including the unseen one) only on the hit segments. We extracted pairwise comparisons from all the segment annotations where the segment of the unseen system was a hit and computed the overall score only on such extracted comparisons.

The results of this experiment are given in Table 3. We also report **hit** rate, which is the ratio of hits to all relevant segments (miss + hits).

The average hit rate is 58.8 %, which is above our expectations. However, we see that the hit rate varies considerably across the left-out systems. This is caused by the fact that some systems are very similar; they use the same tools and/or training data. For example all the systems cu-bojar, cu-depfix, cu-funky and both uedin systems are based on the Moses SMT toolkit and their hit rates are very high (0.74–0.93).

As apparent from the table, the obtained orderings of the systems are not very good. The winning system in each of the tables is the one that was left out, which is obviously wrong. Besides that, systems similar to the left-out one get also a much better rank. For example, when the left-out system is one of the systems cu-bojar, cu-depfix and cu-funky, the other two of this group are right below the left-out system. This could be explained by the following statement: MT systems are more likely to agree on a good translation of a segment than on a bad translation. Our technique

system	score
uedin-uncnstr.	0.633
cu-depfix	0.580
uedin-wmt14	0.576
onlineB	0.571
cu-bojar	0.564
cu-funky	0.560
onlineA	0.499
CU-TectoMT	0.425
commercial1	0.377
commercial2	0.329

(a) uedin-uncnstr., hits: 0.67

system	score
commercial1	0.581
uedin-uncnstr.	0.557
onlineB	0.552
uedin-wmt14	0.540
cu-depfix	0.535
cu-funky	0.525
cu-bojar	0.523
onlineA	0.472
CU-TectoMT	0.422
commercial2	0.346

(b) commercial1, hits: 0.28

system	score
commercial2	0.570
onlineB	0.552
uedin-uncnstr.	0.548
cu-depfix	0.535
cu-bojar	0.529
cu-funky	0.524
uedin-wmt14	0.523
onlineA	0.457
CU-TectoMT	0.423
commercial1	0.386

(c) commercial2, hits: 0.28

system	score
CU-TectoMT	0.649
cu-depfix	0.600
cu-bojar	0.584
cu-funky	0.575
onlineB	0.528
uedin-uncnstr.	0.522
uedin-wmt14	0.502
onlineA	0.450
commercial1	0.373
commercial2	0.339

(d) CU-TectoMT, hits: 0.45

system	score
onlineB	0.689
uedin-uncnstr.	0.584
cu-depfix	0.578
cu-funky	0.567
cu-bojar	0.567
uedin-wmt14	0.564
onlineA	0.511
CU-TectoMT	0.406
commercial1	0.360
commercial2	0.320

(e) onlineB, hits: 0.52

system	score
onlineA	0.649
onlineB	0.584
uedin-uncnstr.	0.577
uedin-wmt14	0.568
cu-depfix	0.566
cu-bojar	0.555
cu-funky	0.546
CU-TectoMT	0.411
commercial1	0.365
commercial2	0.318

(f) onlineA, hits: 0.47

system	score
cu-funky	0.630
cu-depfix	0.600
cu-bojar	0.582
uedin-uncnstr.	0.559
onlineB	0.557
uedin-wmt14	0.542
onlineA	0.491
CU-TectoMT	0.446
commercial1	0.378
commercial2	0.331

(g) cu-funky, hits: 0.74

system	score
cu-bojar	0.588
cu-depfix	0.582
cu-funky	0.564
onlineB	0.562
uedin-uncnstr.	0.559
uedin-wmt14	0.545
onlineA	0.498
CU-TectoMT	0.449
commercial1	0.392
commercial2	0.346

(h) cu-bojar, hits: 0.88

system	score
cu-depfix	0.587
cu-bojar	0.570
cu-funky	0.566
onlineB	0.562
uedin-uncnstr.	0.561
uedin-wmt14	0.548
onlineA	0.498
CU-TectoMT	0.449
commercial1	0.394
commercial2	0.345

(i) cu-depfix, hits: 0.93

Table 3: The results of evaluating unseen systems using exact matching and the **leave-one-out** technique. Each subtable is labelled with the left-out system. We also report the hit rates. The table for the system uedin-wmt14 is omitted for the sake of brevity.

thus faces a sparse data issue which affects wrongly translated spans much more than good ones.

To support this statement we have performed an analysis of some sentences from the test set. In the following example sentences, we have marked the hit segments by **bold font** and the missed segments by *italic font*:

Source: *Amongst other things*, it showed that the Americans even monitored **the mobile phone** *of German Chancellor Angela Merkel*.

Candidate: *Kromě jiných věcí* ukázalo, že Američané i sledovali **mobilní telefon** *Germana kancléře Angely Merkelové*.

The missed segment "Kromě jiných věcí" in this sentence is translated quite well (so the above statement does not hold here). However, the only hit segment here "mobilní telefon" is translated correctly and the missed segment "Germana kancléře Angely Merkelové" is not translated correctly. Judging how easy a segment is to translate, is even more difficult than judging the translations. Nevertheless it is obvious that the hit segment "the mobile phone" is easy to translate and the missed segment "of German Chancellor Angela Merkel" is much more difficult to translate. We can see this also in the last example:

Source: They had *searched frantically for their missing dog* and posted appeals **on social networking sites** after she had ran **into the quarry** *following the minor accident*.

Candidate: Měli zoufale *hledal své chybějící psa* a odvolání **na sociálních sítích** poté, co se dostal **do lomu** *po drobné nehody*.

Both of the hit segments are translated very well and we can say that they are also very easy to translate. On the other hand, the missed segments are understandable but not perfect. Compared to the hit segment, they are also more difficult to translate.

Following the manual analysis, we can conclude that MT systems are much more likely to agree on better translations. This however prevents us from matching the segments exactly, because it gives us very overestimated scores for the unseen candidates.

4.2. Matching the Closest Segment by Edit Distance

We would like to compute the scores for all the segments to avoid the problem stated in the previous section. A natural way to approximate the "correct" ranks for unseen segments is to use the rank of the segment from the database with the closest edit distance. We use the character-based Levenshtein distance, which is defined as

the minimum number of insertions, deletions and substitutions of characters required to transform a given string to another:

$$\mathrm{lev}(a, b) = \min_{e \in E(a,b)} \big(D(e) + S(e) + I(e)\big)$$

where $E(a, b)$ denotes a set of all sequences of edit operations that transform the string a to the string b, and $D(e)$, $S(e)$, $I(e)$ denote number of deletions, substitutions and insertions respectively in the sequence s. If more segments in the database have the same minimal distance to the unseen segment, we compute the average of their ranks.

Similarly to the previous experiment, we extracted the pairwise comparisons and computed the **Ratio of wins (ignoring ties)**, see Table 4 for the results. For each left-out system, we also report the average edit distance (AED) of the unseen segments to the closest segments in the database.

The overall rankings of the systems are much more reasonable compared to the exact matching, although they are still not perfect. The scores of systems that were left out are not always the best in the obtained rankings but they are still heavily overestimated. This shows not only that systems are more likely to agree on the better translations than on the worse ones, but also that they produce translations that are closer to better translations than to other translations of similar quality.

The average edit distances vary a lot. The systems cu-bojar, cu-depfix and cu-funky have very low AED (0.2–1.7), because they are very similar to each other. Systems CU-TectoMT, onlineA and onlineB are in the middle of the AED range (3.1–3.9) and since they are quite solitary in the set of ranked systems we can consider their AEDs as representative values. Systems commercial1 and commercial2 have the highest AEDs. This could be explained by the fact that both of the systems are rule based and produce dissimilar translations to those generated by the statistical based systems. It is interesting, however, that their translations are not even similar to each other.

To support the above statement (that the closest segment to an unseen candidate is more likely to be of better quality), we have performed the following analysis: For each left-out system we computed how often the closest segment has better, equal or worse rank than the "unseen" segment. (We can actually do that, because we know the true rank of the segment before it was removed from the database.) We computed these relative frequencies only on the missed segments (the closest segment was not the same segment). The outcomes for the individual systems are given in Table 5. In total, more than a half of the closest segments have a better rank than the original segments and only 20.6 % of the segments have the same rank. This is a very poor result because it means that our approximation method that ranks unseen translations has the accuracy of only 20.6 %. This accuracy does not differ much for individual systems but there is an expected trend of similar systems (cu-bojar, cu-depfix) having this accuracy slightly higher.

Figure 3 plots how these relative frequencies vary with the edit distance. We see that the relative number of closest systems that are ranked better than the original

system	score
uedin-uncnstr.	0.592
cu-depfix	0.576
onlineB	0.562
cu-bojar	0.558
cu-funky	0.555
uedin-wmt14	0.548
onlineA	0.498
CU-TectoMT	0.446
commercial1	0.397
commercial2	0.347

(a) uedin-uncnstr., AED: 2.0

system	score
cu-depfix	0.566
onlineB	0.553
uedin-uncnstr.	0.551
cu-bojar	0.548
cu-funky	0.545
uedin-wmt14	0.537
commercial1	0.495
onlineA	0.488
CU-TectoMT	0.433
commercial2	0.334

(b) commercial1, AED: 5.4

system	score
cu-depfix	0.559
onlineB	0.549
uedin-uncnstr.	0.546
cu-bojar	0.541
cu-funky	0.539
uedin-wmt14	0.533
commercial2	0.484
onlineA	0.483
CU-TectoMT	0.430
commercial1	0.379

(c) commercial2, AED: 5.7

system	score
cu-depfix	0.566
CU-TectoMT	0.557
onlineB	0.554
uedin-uncnstr.	0.552
cu-bojar	0.548
cu-funky	0.545
uedin-wmt14	0.539
onlineA	0.489
commercial1	0.387
commercial2	0.337

(d) CU-TectoMT, AED: 3.9

system	score
onlineB	0.614
cu-depfix	0.574
uedin-uncnstr.	0.559
cu-bojar	0.556
cu-funky	0.552
uedin-wmt14	0.546
onlineA	0.496
CU-TectoMT	0.444
commercial1	0.395
commercial2	0.345

(e) onlineB, AED: 3.1

system	score
onlineA	0.581
cu-depfix	0.570
onlineB	0.556
uedin-uncnstr.	0.555
cu-bojar	0.553
cu-funky	0.549
uedin-wmt14	0.542
CU-TectoMT	0.441
commercial1	0.391
commercial2	0.341

(f) onlineA, AED: 3.4

system	score
cu-funky	0.597
cu-depfix	0.575
onlineB	0.561
uedin-uncnstr.	0.559
cu-bojar	0.557
uedin-wmt14	0.546
onlineA	0.497
CU-TectoMT	0.445
commercial1	0.396
commercial2	0.346

(g) cu-funky, AED: 1.7

system	score
cu-bojar	0.580
cu-depfix	0.576
onlineB	0.562
uedin-uncnstr.	0.561
cu-funky	0.555
uedin-wmt14	0.548
onlineA	0.499
CU-TectoMT	0.447
commercial1	0.398
commercial2	0.348

(h) cu-bojar, AED: 0.4

system	score
cu-depfix	0.579
onlineB	0.564
uedin-uncnstr.	0.563
cu-bojar	0.561
cu-funky	0.556
uedin-wmt14	0.550
onlineA	0.501
CU-TectoMT	0.448
commercial1	0.399
commercial2	0.349

(i) cu-depfix, AED: 0.2

Table 4: The results of evaluating unseen systems using edit distance matching and **leave-one-out** technique. Each subtable is labelled with the left-out system. The abbreviation AED stands for average edit distance, which is computed across all segments. The table for the system uedin-wmt14 is omitted for the sake of brevity.

Unseen system	Worse	Equal	Better
commercial1	28.0 %	18.2 %	53.8 %
commercial2	23.4 %	16.8 %	59.8 %
cu-bojar	22.8 %	30.5 %	46.7 %
cu-depfix	34.2 %	31.4 %	34.4 %
cu-funky	29.3 %	22.9 %	47.8 %
CU-TectoMT	26.2 %	17.8 %	56.0 %
onlineA	28.7 %	19.1 %	52.2 %
onlineB	33.5 %	19.9 %	46.6 %
uedin-unconstrained	32.8 %	21.5 %	45.7 %
uedin-wmt14	32.1 %	21.9 %	46.0 %
All	28.5 %	19.7 %	51.9 %

Table 5: Comparisons of the unseen and the closest segments' ranks. This table shows how often the rank of the closest segment in the database was worse, equal or better than the original rank of the "unseen" segment. These relative frequencies were computed only on the missed segments (which weren't already in the database).

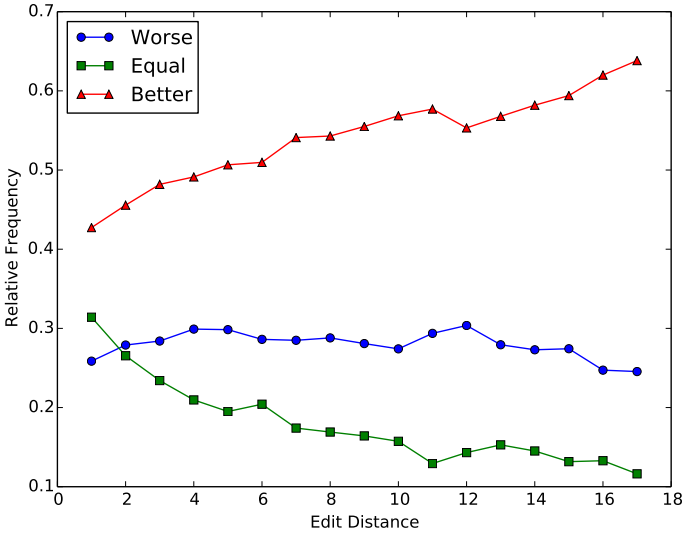


Figure 3: Comparisons of the unseen and the closest segments' ranks with respect to the edit distance. The results in this figure are computed on all the systems.

Unseen segment	Closest segment	D	C
s dokumenty upřednostňujícím doprovodem hradu	se dokumenty favorizovat doprovod hradu	18	**W**
ze 120 domova m2	z 120 m2 domova	7	**B**
vaše ústa ohřívá protein	vaše ústa zahřívání bílkovin	10	**B**
videokonference	videokonferenci	1	**B**
popřel užívání kokainu a	popřela užívání kokainu a	1	**W**
přibližně šedesát- drah kilometru	přibližně šedesát-dráha kilometru	3	**E**
v Liverpoolském porotním soudu	Liverpool Korunního soudu	11	**B**
je nesmysl v gravitaci filmu	Je nesmysl ve filmu gravitace	15	**B**
Pak 64,23 % oprávněných voličů	pak 64,23 % oprávněných voličů	1	**B**
podle DPA agentury	podle DPA agentury té	3	**W**

Table 6: Example candidates and the closest segments. The second last column (**D**) stands for distance and contains distances of the unseen candidates to the closest segments. The last column (**C**) stands for comparison; the closest segment is either worse (**W**), equally good (**E**) or better (**B**) than the original unseen segment.

segment grows quite significantly with the edit distance. The relative number of the worse segments is more or less stable (around 0.3), independent of the edit distance. The relative number of closest segments that are ranked equally as the original segment is decreasing with the edit distance. For example, for the segments whose edit distance to the closest segment is 17, only 10 percent of the closest segments are equally ranked. Relying on similar segments to predict the quality of a new one thus cannot work.

We also list a few example candidate segments in Table 6 together with the corresponding closest segments from the database and their distances. The last column in the table indicates whether the closest segment was ranked better, equal or worse than the "unseen" one.

Unfortunately, we have to conclude that the proposed method, which reuses the database for evaluating unseen translations, does not work. We analyzed the results and the main cause of this failure seems to be that the systems tend to agree on better translations and their translations tend to be more similar to better translations in the database so we cannot predict their rank accurately.

4.3. Enhancing Reference Translations

Following the conclusions from the previous two sections, it seems that errors in machine translation are very unique. Any database of bad examples (bad translations) is therefore very sparse.

In the following experiment, we therefore use only the good candidate segments from the annotated database. The approach used here is, however, different from the previous experiments. We would like to measure how similar candidates are to the good translations from the database. This resembles what automatic metrics do when measuring the similarity between a candidate and reference translations. We have therefore decided to use one of the standard metrics – BLEU – to measure this similarity. First, we are going to introduce the metric and then we are going to customize it for our experiment.

Metric BLEU was developed by Papineni et al. (2002) and it is one of the most popular metrics in the machine translation evaluation. It is defined as the geometric mean of n-gram precisions for $n \in \{1 \ldots N\}$, where N is usually 4. More precisely, for a candidate c and reference translations r_i where $i \in I$, let the clipped count of an n-gram g be defined as follows:

$$\text{count}_{\text{clip}}(g, c, r) = \min\left(\text{count}(g, c),\ \max_{i \in I}(\text{count}(g, r_i))\right)$$

where $\text{count}(g, s)$ denotes the count of n-gram g in the sentence s. The (modified) precision p_n is then defined as:

$$p_n = \frac{\sum_{g \in \text{n-grams}(c)} \text{count}_{\text{clip}}(g, c, r)}{\sum_{g \in \text{n-grams}(c)} \text{count}(g, c)}$$

Using the computed n-gram precisions, we can compute the final BLEU score:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\frac{1}{N}\sum_{i=1}^{N} \log p_n\right)$$

where BP is the brevity penalty (meant to penalize short outputs, to discourage improving precision at the expense of recall) defined as follows:

$$\text{BP} = \begin{cases} 1 & \text{if } |c| > |r| \\ \exp(1 - |r|/|c|) & \text{if } |c| \leq |r| \end{cases}$$

where $|c|$ and $|r|$ are the lengths of the candidate and reference translations respectively. In the case of multiple reference translations, $|r|$ could be the average length, the maximum length or the length closest to the candidate c.

Since the assigned ranks in the database are relative, we cannot know which segments are really good in terms of the absolute quality. We have to assume that there is at least one good candidate translation among the ranked candidates and consider all candidate segments with the best rank as the good translations. We select these candidate segments for each source segment for each sentence.

We use the selected good segments as the reference translations in addition to the original reference sentence translated by a human expert. Since the new references are

System	Score
cu-depfix	0.305
cu-funky	0.302
uedin-unconstrained	0.302
cu-bojar	0.300
uedin-wmt14	0.296
onlineB	0.289
onlineA	0.259
CU-TectoMT	0.225
commercial1	0.176
commercial2	0.160

(a) SegRanksBLEU, correlation: 0.9745

System	Score
cu-depfix	0.221
cu-bojar	0.221
cu-funky	0.221
uedin-unconstrained	0.220
uedin-wmt14	0.215
onlineB	0.207
onlineA	0.187
CU-TectoMT	0.157
commercial1	0.114
commercial2	0.102

(b) BLEU, correlation: 0.9751

Table 7: Overall system ranking according to SegRanksBLEU and BLEU scores. For both of the metrics, we have computed Pearson correlation with human scores.

only short segments and do not cover a whole sentence, we use only the length of the original reference sentence in the computation of the brevity penalty. To distinguish this method from the standard BLEU with the single official reference translation, we will call this method SegRanksBLEU.

Please note, that introducing the new reference translations, which do not change the brevity penalty, can only increase the clipped counts of n-grams occurring in the short segments. Candidates will be rewarded for having n-grams that are also in the good segment translations in the database.

The overall rankings of the evaluated systems as given by SegRanksBLEU and BLEU are in Table 7. As expected, the SegRanksBLEU scores are indeed much higher than BLEU. However, the reported system level correlations of these two metrics are almost equal (correlation of SegRanksBLEU is even a little bit lower). The additional "reference translations" of short segments thus do not bring any useful information.

5. Experiments with MERT

The MERT (Minimum Error Rate Training) method is most often used with BLEU. Recently, there have been a lot of experiments utilizing other automatic metrics. In WMT11 (Callison-Burch et al., 2011), there was a shared task, in which participants tuned a common system with different metrics, and WMT15 plans to run the task again[4]. In WMT11, the tuned systems were then evaluated by humans and some of them outperformed the baseline system tuned with BLEU.

[4] http://www.statmt.org/wmt15/tuning-task/

It is not feasible to employ any sort of human evaluation directly in the MERT process. On one hand, human evaluation is very slow and expensive, on the other hand, MERT requires to evaluate very long n-best list in each iteration. There are some suggestions to do the manual evaluation in a clever way and lower the amount of manual work. For example, Zaidan and Callison-Burch (2009) noticed that a lot of short segments are repeated in a n-best list and therefore suggest to extract these short segments from a n-best list in each MERT iteration and let humans rank them. (Actually our short segment extraction method was partially inspired by this work.) However, they did not try this method in an actual MERT run yet for lack of resources.

We believe that a much less expensive way to introduce an element of human evaluation in the MERT method is to use some sort of semi-automatic metric in which a certain amount of manual work is needed at the beginning and then the metric evaluates translations automatically. In this section, we therefore experiment with previously introduced metrics, which rely on the collected database of short segment ranks.

5.1. Tuned System

The system we tried to tune is the system cu-bojar by Tamchyna et al. (2014), which we also used in previous sections as one of the evaluated systems. This system is Moses-based and combines several different approaches:

- factored phrase-based Moses model
- domain-adapted language model
- deep-syntactic MT system TectoMT

The parallel data used to train the phrase tables consist of 14.83 million parallel sentences taken from the CzEng corpus (Bojar et al., 2012) and 0.65 million sentences taken from the Europarl corpus (Koehn, 2005). The monolingual data used to train language models consist of 215.93 million sentences taken from the Czech side of the CzEng corpus and from 5 sources of a news domain.

We used the implementation of MERT that is distributed with Moses toolkit.[5]

5.2. Metric Variants

Our original idea was that we will use the alignment produced by the Moses decoder when translating the n-best list to project the extracted short source segments to the target side to have candidates that would be extracted the same way as the ranked segments in the database. Unfortunately, the alignment produced by the Moses decoder is very sparse and unreliable (which may be caused by a bug in the code), so we had to get along without the alignment.

The naive approximation is to just test whether the ranked candidate segments from the database also occur in evaluated sentences. The first metric we use in the

[5] http://www.statmt.org/moses/

MERT experiment is therefore very similar to the **Exact Matching** method in Subsection 4.1. For each ranked source segment we test if any of its candidate segments occurs in the evaluated sentence. If it does, we extract all the pairwise comparisons from the matched segment. If more candidate segments occur in the evaluated translation, we choose the longest segment (assuming that the shorter segments are just substrings of the longest one). The final score is then computed as **Ratio of wins (ignoring ties)**. We call this metric ExactMatch in the following.

Even if the hit rate (percentage of segment candidates that are already ranked in the database) computed on a whole n-best list would be 100 %, the ExactMatch metric still does not evaluate whole sentences and could be too coarse and harsh. Moreover, if the hit rate drops during the tuning, the metric score would be computed on a very small percentage of the development set and it would be very unstable. To ensure that the tuning is stable, we interpolate all the metrics in this section (if not said otherwise) together with BLEU with equal weights. We also tried to tune using the ExactMatch metric solely but the tuned system translated very badly and the hit rate dropped very low during the tuning. We could explain this by a hypothesis that the system was tuned to translate a few ranked segments well (so these segments were hits) but other segments were missed and translated badly. The optimal metric score was therefore computed on a very small fraction of the development set and did not reflect the overall quality of the translation.

Since we do not have the alignment and cannot extract the candidate segments exactly, we unfortunately cannot use the method introduced in Subsection 4.2, which matches the closest segment in the database by edit distance. To avoid the shortcomings of the ExactMatch metric (the metric is not computed on all the extracted source segments and an unseen system is more likely to cause a hit in the database with better translations), we propose another variant called PenalizeUnknown. This variant differs from ExactMatch in that it considers all missed segments as the worst translations. We agree that the assumption that all unseen and unranked segments are wrong is not correct, but this approach could increase the hit rate. The question is then whether we prefer to have a system that produces a lot of segments which were already ranked (even badly) or a system that produces a lot of unranked segments, which we hope to be of better quality.

The last variant we experiment with is SegRanksBLEU metric introduced in Subsection 4.3. Because this metric is already based on BLEU metric (and uses the reference translation) we do not interpolate it with BLEU anymore.

5.3. Results and Analysis

We report the results of the tuned system in Table 8. We used the tuned systems to translate the test set (newstest2012) and then we evaluated these translations automatically and manually. For the automatic evaluation we used metrics BLEU (Papineni et al., 2002) and CDER (Leusch et al., 2006). We have also conducted a small scale

Tunable metric	#Mert iterations	Automatic Evaluation BLEU	1-CDER	Manual Evaluation Better	Worse
BLEU (baseline)	11	**0.1782**	**0.3855**	—	—
ExactMatch	20	0.1637	0.3674	22 %	38 %
PenalizeUnknown	8	0.1772	0.3850	**34 %**	**25 %**
SegRanksBLEU	8	0.1753	0.3835	29 %	49 %

Table 8: Results of systems that were optimized to a SegRanks based metric. The items in the first column specify the metric that was used when tuning the system on the development test. The columns BLEU and 1-CDER contain just scores of these metrics computed on the test set translated with the tuned weights. The last two columns contain percentages of better and worse sentences compared to the baseline system in the manual evaluation.

manual evaluation. For each evaluated system, we randomly sampled 100 sentences which were translated differently[6] to the baseline and by the evaluated system. Then we compared manually the sampled sentences with the corresponding translations produced by the baseline system. The task was to choose which sentence is better or whether they are of the same quality. We report how many of the sampled sentences were better and how many of them were worse than the corresponding baseline translation.

None of the metric variants outperformed the baseline system tuned solely to BLEU in the automatic evaluation. This was expected, since the best performing system according to BLEU should be the one that was tuned by BLEU. However, the system tuned by PenalizeUnknown has both the BLEU and CDER scores only a little bit lower.

In the manual evaluation, the only system that outperformed the baseline was the system tuned to PenalizeUnknown. This means that forcing the system to produce known and evaluated segments when translating development set helps to chose better weights. This, however, also means that we discouraged the tuned system to produce unknown and maybe better translations. The best hypothetical translation according to the optimized metric that the tuned system can produce during MERT consists of the best ranked segments from the database. However, there certainly exist better translations.

To see the differences between ExactMatch and PenalizeUnknown, we have plotted the values of hit rates computed in each MERT iteration in Figure 4. PenalizeUnknown gets to the hit rate of 0.7 very quickly (in the sixth iteration) and it's growth is

[6] 58.6 % of all test set sentences were translated differently to the baseline by SegRanksBLEU, 62.4 % by PenalizeUnknown and 83.4 % by ExactMatch

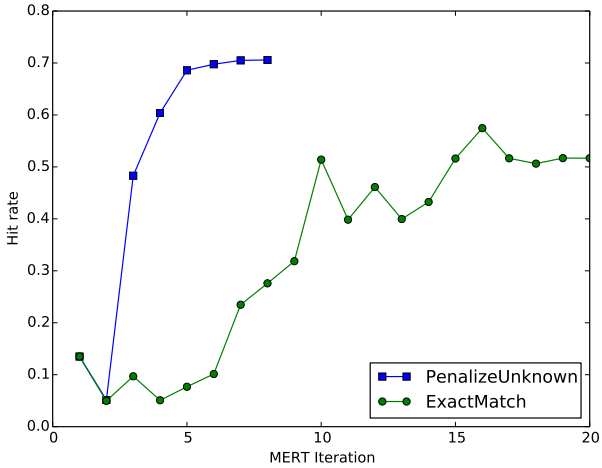


Figure 4: Hit rates computed on the n-best lists produced in each MERT iteration

quite stable. This is expected, since the objective function also indirectly optimizes the hit rate. The hit rate in ExactMatch also grows but much more slowly. It stabilizes around the of value 0.5. It is good that the final value is quite high so the ExactMatch is computed on a non-negligible amount of data. However, it takes many more iterations for ExactMatch to get to the value of 0.5 that it takes for PenalizeUnknown to get to the value of 0.7.

6. Conclusions

In this article, we proposed a new method for manual MT evaluation, called **SegRanks**, in which the annotators rank short segments of a translated sentence relative to each other. Ranking of short segments is easier than ranking of whole sentences and therefore faster. Furthermore, we have developed an easy-to-use and modern annotation interface and conducted a manual evaluation experiment using the proposed method.

When computing overall system scores, the short segment ranks give similar results to the official WMT evaluation with less effort. The measured inter- and intra-annotator κ scores (the normalized agreements) are indeed slightly higher than the corresponding values in the WMT manual evaluation.

To explore the possibility of reusing the collected database of ranked segments to evaluate unseen translations, we have performed several experiments. Although

most of them did not work as expected, we tried to identify and analyze the roots of the failures. The main cause seems to be the fact that each error in machine translation is unique and that segments produced by more than one system are likely to be of better quality. This is also related to another observation we make, that translations are more likely to be closer to better translations than to translations equally good or worse. Maybe, if we had a more dense database (many more than 10 evaluated systems), these phenomena would not influence the results so adversely.

In the last experiment, we tried to use the database to tune a machine translation system using the MERT method. We proposed several variants of **SegRanks**-based metrics adapted for the MERT tuning. The tuned systems were evaluated by humans against the baseline system tuned by BLEU. We were able to improve the tuning of the system using the technique that considered unseen segments as bad and therefore pushed the system to produce known and already evaluated segments.

Bibliography

Bojar, Ondřej, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. The Joy of Parallelism with CzEng 1.0. In *Proceedings of LREC2012*, Istanbul, Turkey, May 2012. ELRA, European Language Resources Association. In print.

Bojar, Ondřej, Matouš Macháček, Aleš Tamchyna, and Daniel Zeman. Scratching the surface of possible translations. In *Text, Speech, and Dialogue*, pages 465–474. Springer, 2013.

Bojar, Ondřej, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 2014.

Callison-Burch, Chris, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (Meta-) Evaluation of Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W07-0718`.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, and Omar Zaidan. Findings of the 2011 Workshop on Statistical Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W11-2103`.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June 2012. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W12-3102`.

Cohen, J. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37, 1960.

Klein, Dan and Christopher D. Manning. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003a. Association for Computational Linguistics. doi: 10.3115/1075096.1075150. URL `http://dx.doi.org/10.3115/1075096.1075150`.

Klein, Dan and Christopher D. Manning. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS)*, pages 3–10. MIT Press, 2003b.

Koehn, Philipp. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT. URL `http://mt-archive.info/MTS-2005-Koehn.pdf`.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073462. URL `http://dx.doi.org/10.3115/1073445.1073462`.

Landis, J Richard and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

Leusch, Gregor, Nicola Ueffing, and Hermann Ney. CDER: Efficient MT Evaluation Using Block Movements. In *In Proceedings of EACL*, pages 241–248, 2006.

Och, Franz Josef and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Comput. Linguist.*, 29(1):19–51, Mar. 2003. ISSN 0891-2017. doi: 10.1162/089120103321337421. URL http://dx.doi.org/10.1162/089120103321337421.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL the 40th annual meeting on Association for Computational Linguistics*, pages 311–318, 2002.

Tamchyna, Aleš, Martin Popel, Rudolf Rosa, and Ondrej Bojar. CUNI in WMT14: Chimera Still Awaits Bellerophon. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 195–200, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3322.

Zaidan, Omar F. and Chris Callison-Burch. Feasibility of Human-in-the-loop Minimum Error Rate Training. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 52–61, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-59-6. URL http://dl.acm.org/citation.cfm?id=1699510.1699518.

Address for correspondence:
Ondřej Bojar
bojar@ufal.mff.cuni.cz
Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics,
Charles University in Prague
Malostranské náměstí 25
118 00 Praha 1, Czech Republic

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015 111–130

Ultrametric Distance in Syntax

Mark D. Roberts

Charles University in Prague, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics

Abstract

Phrase structure trees have a hierarchical structure. In many subjects, most notably in **taxonomy** such tree structures have been studied using ultrametrics. Here syntactical hierarchical phrase trees are subject to a similar analysis, which is much simpler as the branching structure is more readily discernible and switched. The ambiguity of which branching height to choose, is resolved by postulating that branching occurs at the lowest height available. An ultrametric produces a measure of the complexity of sentences: presumably the complexity of sentences increases as a language is acquired so that this can be tested. All ultrametric triangles are equilateral or isosceles. Here it is shown that $\overline{X}$ structure implies that there are no equilateral triangles. Restricting attention to simple syntax a minimum ultrametric distance between lexical categories is calculated. A matrix constructed from this ultrametric distance is shown to be different than the matrix obtained from features. It is shown that the definition of C-COMMAND can be replaced by an equivalent ultrametric definition. The new definition invokes a minimum distance between nodes and this is more aesthetically satisfying than previous varieties of definitions. From the new definition of C-COMMAND follows a new definition of of the central notion in syntax namely GOVERNMENT.

1. Introduction

1.1. Ultrametric Literature

Ultrametrics are used to model any system that can be represented by a bifurcating hierarchical tree. To list briefly some areas where ultrametrics have been applied. Perhaps the most important application is to taxonomy, Jardine and Sibson (1971, Ch.7), and Sneath and Sokal (1973). Here the end of a branch of the tree represents a species and the ultrametric distance between them show how closely the species

 Corresponding author: robemark@gmail
Cite as: Mark D. Roberts. Ultrametric Distance in Syntax. The Prague Bulletin of Mathematical Linguistics No. 103, 2015, pp. 111–130. doi: 10.1515/pralin-2015-0006.

are related. Hierarchical cluster methods classify species and also shows how closely species are related. This technique has also been used in semantics, Shepard and Arabie (1979). The technique can become quite complicated because it involves statistical analysis with continuous variates. Ultrametrics have been applied frequently in the theory of spin glass, Weissman (1993). Ultrametrics have been used for description of slowly driven dissipative systems, which exhibit avalanche-like behaviour, these include earthquakes, extinction events in biological evolution, and landscape formation, Boettcher and Paczuski (1997); also ultrametrics can describe systems with fast relaxation, Vlad (1994). Ultrametrics are used in the theory of neural nets, Parga and Virasoro (1986). The dynamics of random walks on ultrametric spaces have been studied, Ogielchi and Stein (1985). Ultrametrics have been applied to the thermodynamics of macromolecules such as RNA, Higgs (1996), the directed polymer problem Perlman and Schwarz (1992), and sociology Schweinberger and Snijders (2003). Bounds on the size of ultrametric structure have been discussed by Baldi and Baum (1986). From a more theoretical angle, a category theory approach has been elucidated by Rutten (1996), a model theoretic approach to ultrametrics is given by Delon (1984), and ultrametric might be related to T-theory, Andreas Dress and Terhalle (1996). The relationship between ultrametric distance and hierarchy is further discussed in Guénoche (1997). Construction of optimal ultrametric trees is discussed by Young and DeSarbo (1995). Ultrametrics are related to p-adelic quantities, Karwowski and Mendes (1994), Murtagh (2004) and B. Dragovich and Volovich (2009). P-adelic quantities are used in string theory: the way that ultrametrics enters here is explained in §10&§13.4 of Brekke and Freund (1993). There does not seem to be any straightforward connection of any of the above to the optimization techniques of Prince and Smolensky (1997). As well as ultrametric trees, there are also *decision trees* Hammer (1998), and the connection between them is still not known. Some of the above ultrametric applications have been reviewed by R. Rammal and Virasoro (1986). Chomskian syntatic trees have been axiomatized by Backofen and Vijay-Shankar (1995).

1.2. Ultrametric Inequalities

There is the following relationship between trees and ultrametrics. An N-leaf edge(node)-weighted tree corresponds to an $N \times N$ square matrix M in which $M_{ij} =$ the sum of the weights of the edges (nodes) in the shortest path between i and j. When the weights are non-zero and non-negative, M is a distance in the usual sense.

$$\forall x, y, z \quad M_{xy} = 0 \quad \text{if } x = y \tag{1}$$

$$M_{xy} > 0 \quad \text{for} \quad x \neq y \tag{2}$$

$$M_{xy} = M_{yx} \tag{3}$$

$$M_{xy} \leq M_{xz} + M_{zy}. \tag{4}$$

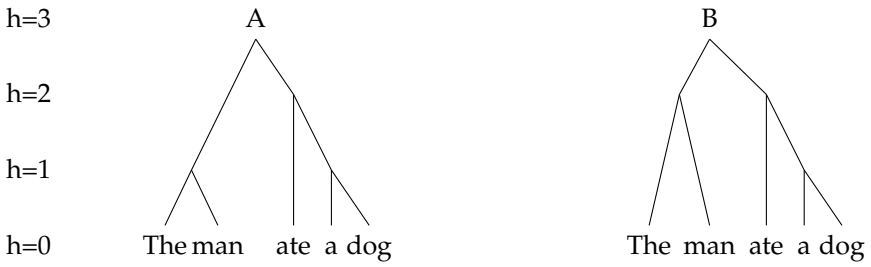


Figure 1: Different syntactic descriptions of "the man ate a dog"

However if the triangle inequality (4) is replaced by

$$M_{xy} \leq \max\{M_{xz}, M_{zy}\}. \tag{5}$$

then M is an ultrametric, equation (5) implies (4).

1.3. Syntactic Phrase Trees

For the analysis of syntactic phrase trees the necessary technique is quite simple and is illustrated by the examples in Section 2. Psychological analysis of phrase trees has been carried out by Johnson (1965) and Levelt (1970). The phrase tree approach has critics Botha (1965); also Evans and Levison (2009) question the existence of language universals. The examples here mainly follow the examples in Lockward (1972), Kayne (1981), McCloskey (1988), and especially Haegeman (1994). There are at least five reasons for introducing an ultrametric description of syntax.

The first is to completely specify tree (also called *dendrogram*) structure. Consider the following example illustrated by Figure 1.

For current syntactic models sometimes nodes are taken to occur at the highest level, and sometimes the two trees are equivalent see McCloskey (1988, footnote 6); however consider the ultrametric distance between 'the' and 'man',

$$A(the, man) = 1, \qquad B(the, man) = 2, \tag{6}$$

where the numbers are the height of the lowest common node above the two lexical items. This ambiguity does not occur in current syntactic models, and a purpose of an ultrametric model is to disambiguate the difference in height, because this might have consequence in the complexity of the encoded model, see the next point.

The second is it gives a measure of the complexity of a sentence: the greater the ultrametric distance required the more complex a sentence is. The above can also be viewed in terms of 'closeness'. The example Figure 1 illustrates that current syntactic

models give no notion of how 'close' determiners and nouns are. However ultrametrics do give an indication of closeness and this can be compared: *firstly* to the closeness indicated by features, *secondly* to the idea that if elements of a sentence are not sufficiently close then there is a BARRIER Chomsky (1986b) to movement, roughly speaking barriers impede the movement of phrases to different places in a sentence. Only the closeness as indicates by features is looked at here. In traditional syntax phrases can be iteratively embedded to give sentences of unbounded length and complexity. A degree of sentence complexity perhaps corresponds to the height of the tree representing the sentence. As people can only process a finite amount of information this height must be finite. In the traditional theoretical framework there is no finite bound on sentence length. An upper bound could perhaps be found by experiment. Inspection of phrase trees suggests a first guess of $h = 12$.

The third is that it means that syntax is described in the same formalism as that used in a lot of other sciences, for example those topics described in the first paragraph of Section 1.1, so that there is the possibility of techniques being used in one area being deployed in another.

The fourth is that an ultrametric formulation might allow a generalization so that ideas in syntax can be applied to other cognitive processes.

The fifth, see the next section 1.4, and perhaps the most important, is that it might be possible to use some sort of minimum distance principle in syntax: it could be this minimum description which would have application in other cognitive processes. In other words that ultrametric trees should be simple rather than complicated and that the sort of mechanism use to encode simple tress might be used elsewhere.

1.4. Ockham's Razor

Minimum description in science goes back several hundred years to "*Ockham's razor*" or perhaps further, see for example Sorton (1947, page 552). The principle of least action (see for example Bjorken and Drell (1965, §11.2)) in physics is that minimal variation of a given action gives field equations which describe the dynamics of a system. For example, Maxwell's equations can be derived from a simple action by varying it. In the present context one would hope that syntax allows for a minimum encoding of semantic information, the minimum encoding being given by some ultrametric measure. A different approach along these lines is that of Rissanen (1983) and Zadrozny (1999). Briefly they assign a length of 1 to each symbol in a sentence, then the MINIMUM DESCRIPTION LENGTH states that the best theory to explain a set of data is the one which minimizes both the sum of: i) the length, in bits, of the description of the theory, and ii) the length, in bits, of data when encoded with the help of the theory. Christiansen (2001) discusses how constraint handling rules (CHR) can be applied to grammars. This can be thought of as a minimizing procedure.

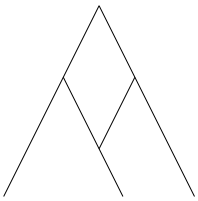


Figure 2: A RETICULATE tree

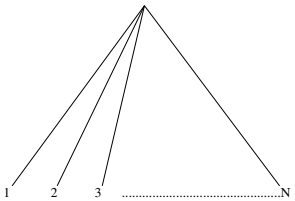


Figure 3: N-ARY branching

1.5. Reticulate & N-ary Trees

A RETICULATE tree is a tree in which there are one or more sets of reconvergent branches, illustrated by Figure 2, a NON-RETICULATE tree is a tree in which the branches do not reconverge. N-ARY branching is illustrated by Figure 3. BINARY branching is N-ARY branching with $N = 2$. N-ARY branching can be replaced by binary branching if additional layers are used. A SWITCHED tree is a tree in which all the branches are binary. Syntactic phrase trees are NON-RETICULATE and SWITCHED. In most linguistic theories all syntactic phrase trees have $\overline{X}$ structure, Jackendoff (1977). $\overline{X}$ structure implies binary branching, see Section 2.3 and Figure 8. Here attention is restricted to theory which has $\overline{X}$ structure.

1.6. Sectional Contents

In Section 2 it is shown how to represent trees by matrices and triangles. All $\overline{X}$ triangles are isosceles but not equilateral. In Section 3 the matrix **U** for the minimum ultrametric distance for lexical categories is given. For simplicity discussion is limited to active voice sentences with only determiners, nouns, transitive verbs, adjectives, and prepositions. Inclusion of case theory, COMP, INFL,.. might be of interest but would complicate matters. In Section 4 the singular matrix **F** for features is given. **F** is not an ultrametric matrix and there appears to be no relation to **U**. In Section 5 it is shown that the notion of C-COMMAND is equivalent to an ultrametric minimum distance. This allows a new definition of government to be given. In appendix Section 7 other linguistic hierarchies are discussed; in particular there appears to be at least two separate occurrences of culturally determined partial ordered hierarchies - the *accessibility hierarchy* for relative clauses and the *universal colour ordering*. For completion in appendix Section 7 there is a very brief account of what these hierarchies are, a comparison and contrasting of them, and the speculation that they are specific examples of a *grand cultural hierarchy*. The question arises of why such hierarchies should exist, and it might be because they reduce the amount of memory needed to process information by clumping information together in the style of Miller (1956); for a more

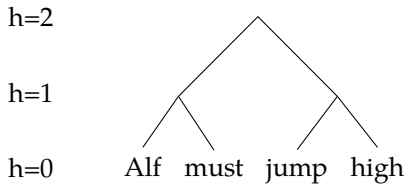


Figure 4: The simplest BINARY tree for "Alf must jump high"

recent reference see Cowan (2001). A *hierarchy* is an example of a representation as discussed by Roberts (2005).

2. $\overline{\mathrm{X}}$ Structure Implies No Equilateral Triangles

2.1. Binary and N-ary Branching for simple sentences

$\overline{\mathrm{X}}$ structure implies BINARY BRANCHING Haegeman (1994, p.139), and the Figure 8. To see what this implies for ultrametric distances consider all five species of BINARY BRANCHED tree. The fixed word order in 'Alf must jump high' reduces the total number of possible matrices form 15 to 8. The *first* has diagram Figure 4 (compare Haegeman (1994, p.141 diagram 84a)) and corresponding matrix:

$$\mathrm{First} = \begin{array}{ccccc} \bullet & \mathrm{A} & \mathrm{M} & \mathrm{J} & \mathrm{H} \\ \mathrm{A} & 0 & 1 & 2 & 2 \\ \mathrm{M} & . & 0 & 2 & 2 \\ \mathrm{J} & . & . & 0 & 1 \\ \mathrm{H} & . & . & . & 0 \end{array} \tag{7}$$

respectively, where A, M, ... are short for 'Alf', 'must'. The matrices corresponding to the other *four* BINARY BRANCHED trees are (compare Haegeman (1994, p.141&142 diagrams 84b,c,d,e):

$$\mathrm{Second} = \begin{array}{ccccc} \bullet & \mathrm{A} & \mathrm{M} & \mathrm{J} & \mathrm{H} \\ \mathrm{A} & 0 & 3 & 3 & 3 \\ \mathrm{M} & . & 0 & 2 & 2 \\ \mathrm{J} & . & . & 0 & 1 \\ \mathrm{H} & . & . & . & 0 \end{array} \tag{8}$$

$$\mathrm{Third} = \begin{array}{ccccc} \bullet & \mathrm{A} & \mathrm{M} & \mathrm{J} & \mathrm{H} \\ \mathrm{A} & 0 & 3 & 3 & 3 \\ \mathrm{M} & . & 0 & 1 & 2 \\ \mathrm{J} & . & . & 0 & 1 \\ \mathrm{H} & . & . & . & 0 \end{array} \tag{9}$$

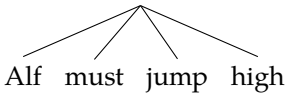


Figure 5: The 4-ary tree for "Alf must jump high"

$$\text{Fourth} = \begin{array}{c|cccc} \bullet & A & M & J & H \\ A & 0 & 2 & 2 & 3 \\ M & . & 0 & 1 & 3 \\ J & . & . & 0 & 3 \\ H & . & . & . & 0 \end{array} \quad (10) \qquad \text{Fifth} = \begin{array}{c|cccc} \bullet & A & M & J & H \\ A & 0 & 1 & 2 & 3 \\ M & . & 0 & 2 & 3 \\ J & . & . & 0 & 3 \\ H & . & . & . & 0 \end{array} \quad (11)$$

There are *two* 3-ary trees with matrices (compare Haegeman (1994) p.142 (Haegeman, 1994) diagrams 84g and 84h): Haegeman (1994, p.142 diagrams 84g and 84h)):

$$\text{Sixth} = \begin{array}{c|cccc} \bullet & A & M & J & H \\ A & 0 & 1 & 1 & 2 \\ M & . & 0 & 1 & 2 \\ J & . & . & 0 & 2 \\ H & . & . & . & 0 \end{array} \quad (12) \qquad \text{Seventh} = \begin{array}{c|cccc} \bullet & A & M & J & H \\ A & 0 & 2 & 2 & 2 \\ M & . & 0 & 1 & 1 \\ J & . & . & 0 & 1 \\ H & . & . & . & 0 \end{array} \quad (13)$$

and finally there is *one* 4-ary tree (compare Haegeman (1994) p.142 (Haegeman, 1994) diagram 84f) Haegeman (1994, p.142 diagram 84f)) with diagram Figure 5 and matrix:

$$\text{Eighth} = \begin{array}{c|cccc} \bullet & A & M & J & H \\ A & 0 & 1 & 1 & 1 \\ M & . & 0 & 1 & 1 \\ J & . & . & 0 & 1 \\ H & . & . & . & 0 \end{array} \quad (14)$$

2.2. Triangle representation of the proceeding matrices

All ultrametric triangles are isosceles with small base, but only some are equilateral. The previous subsection (2.1) suggests that binary branching implies that there are no equilateral triangles in ultrametric models of syntax. For example from matrix (13), $d(A, M) = 1, d(A, J) = 2, d(M, J) = 2$ has the triangle representation Figure 6, and from matrix (14), $d(A, M) = 1, d(A, J) = 1, d(M, J) = 1$ giving in the triangle representation Figure 7.

In the next section it is proved that $\overline{X}$ structure implies that there are no equilateral triangles.

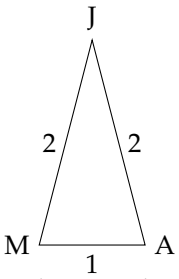


Figure 6: The isosceles triangle

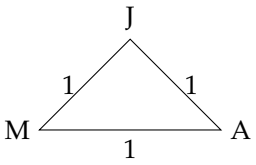


Figure 7: The equilateral triangle

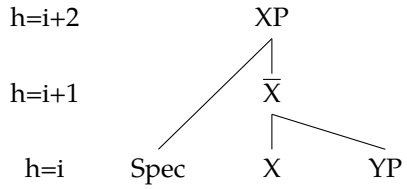


Figure 8: The $\overline{X}$ Template

2.3. The $\overline{X}$ Template

The $\overline{X}$ template Figure 8 is the form that nodes take in syntax.
The matrix representation of this is:

$$\mathbf{X} = \begin{array}{cccc} \bullet & Spec & X & YP \\ Spec & 0 & i+2 & i+2 \\ X & . & 0 & i+1 \\ YP & . & . & 0 \end{array} \tag{15}$$

From this the triangle representation is Figure 9. This is isosceles but not equilateral.

3. The minimum ultrametric distance between lexical categories

3.1. The minimum distance principle

We assume that it is the minimum distance between lexical categories that is important, and refer to this as the MINIMUM DISTANCE PRINCIPLE. In part this is motivated by the discussion in Section 1.4. A current psycholinguistic model of sentence production is the garden path model, see for example Frazier (1987). Part of this model requires *the minimal attachment principle*, which is "do not postulate unnecessary nodes.":

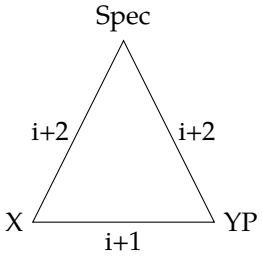


Figure 9: The triangle representation of $\overline{X}$ structure

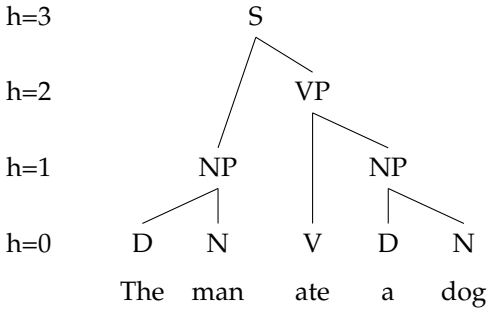


Figure 10: The correct tree for the example in Section 1.

this can be thought of as a minimum principle. The MINIMUM DISTANCE PRINCIPLE implies that the correct tree for equation (6) illustrated by Figure 1 is Figure 10, so that all entries occur at the lowest possible height.

Thus in particular tree **A** is preferred to tree **B**. This assumption does not effect the matrix **U** (16) given and described below, but will have an effect when the analysis is extended to θ-theory, see Haegeman (1994, §3.2.3). From the above $d(N, D) = 1$, $d(N, V) = d(D, V) = 2$. Similarly from Figure 11, $d(V, A) = 4$.

Constructing other examples gives the ultrametric distance matrix

$$\mathbf{U} = \begin{array}{c|cccccc} \bullet & D & N & V & A & P \\ D & 0 & 1 & 2 & 2 & 2 \\ N & . & 0 & 2 & 2 & 2 \\ V & . & . & 0 & 4 & 3 \\ A & . & . & . & 0 & 3 \\ P & . & . & . & . & 0 \end{array} \tag{16}$$

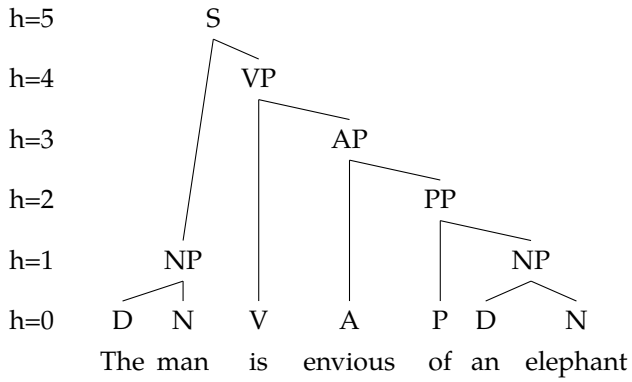


Figure 11: The distance between verbs and adjectives.

Ignoring the determiner D ('a' or 'the') and ordering the matrix NPVA (noun, pronoun, verb, adjective) suggests the pattern

$$\mathbf{I} = \begin{matrix} 0 & i & i & i & \dots \\ . & 0 & i+1 & i+1 & \dots \\ . & . & 0 & i+2 & \dots \\ . & . & . & 0 & \dots \end{matrix} \tag{17}$$

which is compatible with the $\overline{X}$ matrix of the last section; however it does not follow by necessity as the $\overline{X}$ case holds for a single sentence and **U** is constructed from the syntactical representations of several sentences.

4. Features

4.1. A matrix representation of features

This section investigates whether there is a general framework which can describe both the preceding and also "features". An objective is to reduce the large number of objects that can be 'spec' and so on by using the objects features: in other words to introduce a type of atomic theory. Roughly speaking the idea behind features is that adjectives and prepositions have qualities in them that are associated with verbs and nouns, as can be seen from (18) adjectives have +N & +V and prepositions have -N & -V. Haegeman (1994, p.146) gives the following (18) diagram for features:

$$\text{Features diagram} = \begin{matrix} \text{Noun} & +\text{N} & -\text{V} \\ \text{Verb} & -\text{N} & +\text{V} \\ \text{Adj.} & +\text{N} & +\text{V} \\ \text{Pre.} & -\text{N} & -\text{V} \end{matrix} \tag{18}$$

in words nouns have features of +noun and -verb, adjectives have features of +noun and +verb, and so on. This can be represented by the matrix

$$\text{Features matrix} = \mathsf{F} = \begin{array}{cccc} \cdot & \mathit{Noun} & \mathit{Verb} \\ \mathit{Noun} & +1 & -1 \\ \mathit{Verb} & -1 & +1 \\ \mathit{Adj.} & +1 & +1 \\ \mathit{Pre.} & -1 & -1 \end{array} \tag{19}$$

The Pauli matrices (see for example Bjorken and Drell (1965, p.378)) are

$$\mathrm{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ \sigma^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \sigma^2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ \sigma^3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{20}$$

A square matrix can be constructed by assuming that the matrix is symmetric. This leaves only one unknown $\mathsf{F}(A, P)$. Taking $\mathsf{F}(A, P) = -1$ gives equal number of positive and negative entries in the matrix

$$\mathbf{F\ completion} = \mathbf{FC} = \begin{array}{ccccc} \bullet & N & V & A & P \\ N & +1 & -1 & +1 & -1 \\ V & -1 & +1 & +1 & -1 \\ A & +1 & +1 & +1 & -1 \\ P & -1 & -1 & -1 & +1 \end{array} \tag{21}$$

which is singular as its determinant vanishes. There appears to be no relation between matrix **FC** (21) and matrix **U** (16). **FC** can be expressed as

$$\mathbf{FC} = \begin{pmatrix} \mathrm{I} - \sigma^1 & -i\sigma^2 + \sigma^3 \\ +i\sigma^2 + \sigma^3 & \mathrm{I} - \sigma^1 \end{pmatrix}. \tag{22}$$

However this does not correspond in any straightforward way to any of the Dirac matrices (see for example Bjorken and Drell (1965, p.378)) in standard representations.

4.2. Pseudo-inverse of the features matrix

Another way to proceed is to use pseudo-inverses which can be calculated using the program Octave, then

$$\mathbf{pinv}(\mathbf{F}) = \frac{1}{8}\begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 \end{pmatrix}, \tag{23}$$

then in terms of Pauli matrices (20)

$$\mathbf{F} = \begin{pmatrix} \mathrm{I} - \sigma_1 \\ \sigma_3 + i\sigma_2 \end{pmatrix}, \qquad \mathbf{pinv}(\mathbf{F}) = \frac{1}{8}\left(\mathrm{I} - \sigma_1, \sigma_3 - i\sigma_2\right), \tag{24}$$

giving products

$$\mathbf{pinv}(\mathbf{F}) * \mathbf{F} = \mathrm{I}, \qquad \mathbf{F} * \mathbf{pinv}(\mathbf{F}) = \frac{1}{2}\begin{pmatrix} \mathrm{I} - \sigma_1 & 0 \\ 0 & \mathrm{I} - \sigma_1 \end{pmatrix}. \tag{25}$$

Having got an inverse (23) one would hope to be able to use it to predict properties of nouns, verbs and so on, but it is not yet clear how.

4.3. Truth features matrix

The above approach is not unique; consider instead of objects having $\pm N, \pm V$ having truth values

$$\text{Features truth diagram} = \begin{matrix} \text{Noun} & \mathrm{T} & \mathrm{F} \\ \text{Verb} & \mathrm{F} & \mathrm{T} \\ \text{Adj.} & \mathrm{T} & \mathrm{T} \\ \text{Pre.} & \mathrm{F} & \mathrm{F} \end{matrix} \tag{26}$$

with corresponding matrix

$$\text{Features truth matrix} = \mathrm{T} = \begin{matrix} \cdot & \text{Noun} & \text{Verb} \\ \text{Noun} & 1 & 0 \\ \text{Verb} & 0 & 1 \\ \text{Adj.} & 1 & 1 \\ \text{Pre.} & 0 & 0 \end{matrix} \tag{27}$$

with inverse

$$\mathbf{pinv}(\mathbf{T}) = \frac{1}{3}\begin{pmatrix} 2 & -1 & 1 & 0 \\ -1 & 2 & 1 & 0 \end{pmatrix}, \tag{28}$$

giving products

$$\mathbf{pinv}(\mathbf{T}) * \mathbf{T} = \mathrm{I}, \quad \mathbf{T} * \mathbf{pinv}(\mathbf{T}) = \frac{1}{3}\begin{pmatrix} 2 & -1 & 1 & 0 \\ -1 & 2 & 1 & 0 \\ 1 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{29}$$

which do not simplify in terms of Pauli matrices (20).

The structure of the two approaches does not seem to be the same as can be seen by looking at (19) and (27); and so far it is not clear which is better. It is possible that the above features approach can be extended using more features; for example introducing D=determiner one has the $2^3 = 8$ objects $\{\mathrm{D} = (1 \pm 1)/2, \mathrm{N} = (1 \pm 1)/2, V = (1 \pm 1)/2\}$, and one could take $\{1, 0, 0\}$ to be a determiner $\{1, 1, 0\}$ to be a noun phrase and so on, then there is the question of how many features have to be introduced to represent all tree objects.

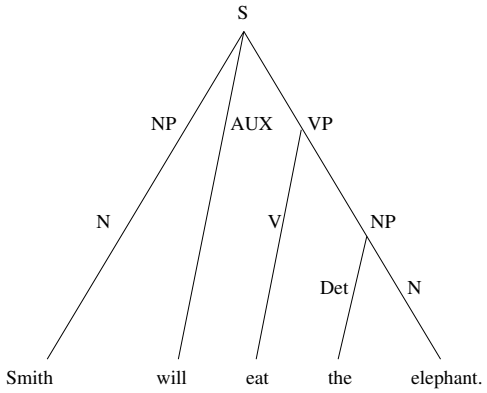


Figure 12: Illustration of DOMINATES

5. Ultrametric Approach to Government

Recall the following definitions in Haegeman (1994):

5.1. Definition of DOMINATES.

Definition Haegeman (1994, p.85) Node **A** DOMINATES node **B** iff:
i) $h(A)$ is higher up or at the same height on the tree as $h(B)$ i.e.$h(A) \geq h(B)$
ii) it is possible to trace a path from **A** to **B** going only downward,
or at most going to one higher node.
Remarks
The *first* requirement is that **A** is at a greater height than **B**. The *second* requirement restricts the possible downward route from **A** to **B** so that it contains at most one upward segment.
Example (compare Haegeman (1994, p.83))The phrase tree in Figure 12 gives the 'dominates' matrix:

$$\mathbf{D} = \begin{array}{c|ccccccccc} \bullet & S & NP(S) & N(S) & AUX & VP & V & NP(E) & Det & N(e) \\ S & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ NP(S) & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ N(S) & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ AUX & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ VP & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ V & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ NP(E) & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ Det & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ N(E) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \tag{30}$$

where 1 indicates "**A** dominates **B**" and 0 indicates that it does not.

5.2. Definition of C-COMMAND

Definition Haegeman (1994, p.134)

Node **A** C-COMMANDS (constituent-commands) node **B** iff:
i) **A** does not dominate **B** and **B** does not dominate **A**,
ii) The first branching node dominating **A** also dominates **B**.

Remarks

The *first* requirement is that there is no direct route up or down from **A** to **B** passing more than one higher node. The *second* requirement restricts **A** and **B** to be 'close'. Haegeman's first criterion for dominance needs to be adjusted: if it is correct then $h(A) > h(B)$ and $h(B) > h(A)$ so that the set of all C-COMMANDS is empty, therefore greater than or equal $\geq$ is used here instead of greater than $>$. Haegeman's second criterion for dominance also needs to be adjusted: if no higher node is allowed the set of C-COMMANDS is again empty. Chomsky (1986a, p.161) approaches the subject in a different manner using maximal projections.

For example, in Figure 13, $0 < j < k < l$. The corresponding ultrametric matrix is

$$\mathbf{U} = \begin{array}{c|cccc} \bullet & A & B & C & D \\ A & 0 & k & k & l \\ B & . & 0 & j & l \\ C & . & . & 0 & l \\ D & . & . & . & 0 \end{array} \tag{31}$$

The C-COMMAND matrix **CM** is

$$\mathbf{CM} = \begin{array}{c|cccc} \bullet & A & B & C & D \\ A & 1 & 1 & 1 & 0 \\ B & 0 & 1 & 1 & 0 \\ C & 0 & 1 & 1 & 0 \\ D & 1 & 1 & 1 & 1 \end{array} \tag{32}$$

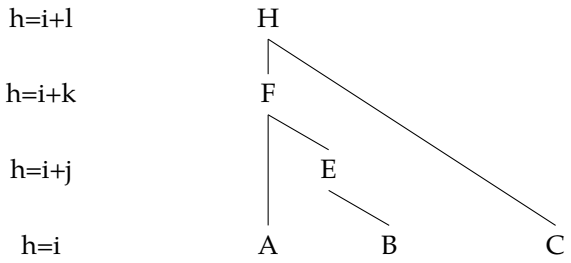


Figure 13: Example of C-COMMANDS.

where 1 indicates **A** C-COMMANDS **A**, and 0 indicates that it does not, similarly for **B, C, D**.

5.3. Definitions of C-DOMAIN & GOVERNS

Definition Haegeman (1994, p.134)
The total of all the nodes C-COMMANDED by an element is the C-DOMAIN of that element.
Definition Haegeman (1994, p.135)
A GOVERNS **B** iff:
i) **A** is a GOVERNOR,
ii)**A** C-COMMANDS **B** and **B** C-COMMANDS **A**.
Remarks:
The *first* requirement is a restriction on the set **A** (in linguistic terminology the category **A**). A GOVERNOR is a part of speech which generalizes the notion of a verb governing an object; unfortunately there does not seem to be a formal definition of it. The *second* requirement is that **A** and **B** should be sufficiently 'close'.

5.4. Definitions of CU-DOMAIN & CU-COMMAND

Now let **D(A)** be the set of all the ultrametric distances to other nodes at the same height and let **M(A)** be the set of these which have the smallest value.
Call **M(A)** the CU-DOMAIN of **A** and say **A** CU-COMMANDS all $\mathbf{B}\varepsilon\mathbf{M}(\mathbf{A})$ (in words **B** is a member of **M(A)**. This is illustrated by Figure 14.

5.5. Theorem showing the identity between C-DOMAIN & CU-DOMAIN

Theorem:
The sets **A** C-COMMANDS **B** and A CU-COMMANDS **B** are identical, likewise the C-DOMAIN and the CU-DOMAIN.
Proof:
From the i) part of the definition of C-COMMAND $h(A) = h(B)$, so that we are only

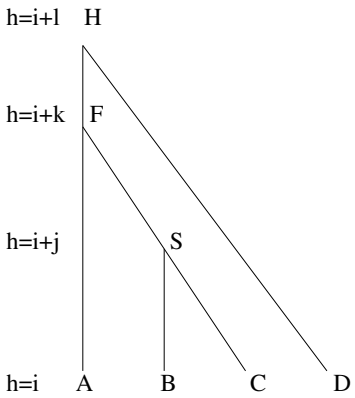


Figure 14: Illustration of the Theorem.

concerned with nodes at the same height $h(A) = i$. Let the first branching node above **A** be **F**, with $h(F) = i+k$. Let **H** be any node dominating **F**, with $h(H) = i+l$. Let **E** be the subsidiary node dominating **B** and **C** and dominated by **F**, with $h(E) = i+j$. The closest nodes to **A** are **B** and **C** both with an ultrametric distance k. The sets **D(A)** and **M(A)** are **D(A)**={**A,B,C,D**}, **M(A)**={**A,B,C**}. **A** both C-COMMANDS and CU-COMMANDS itself and **B** and **C**. The actual integer values $i, j, k, \ldots$ are arbitrary and thus the result holds in general.

5.6. A New Definition of Government

This allows a new definition of GOVERNMENT. **A** GOVERNS **B** iff:
i) **A** is a GOVERNOR.
ii) both $\mathbf{A}\varepsilon\mathbf{M}(\mathbf{B})$ and $\mathbf{B}\varepsilon\mathbf{M}(\mathbf{A})$ (in words **A** is a member of **M(B)** and vice versa). This definition of GOVERNMENT is the same as the previous definition of GOVERNMENT, but with the C-COMMAND requirement replaced by an ultrametric requirement that distances be minimal.

6. Conclusion

The definition of government in Section 5.6 might at sometime in the future allow the five points in Section 1.3 to be addressed.

SU > DO > IO > OBL > GEN > OCOMP

Figure 15: The *accessibility hierarchy*

7. Appendix: Other Linguistic Hierarchies

7.1. The Accessibility Hierarchy

A RELATIVE CLAUSE (*RC*) is a clause that modifies a noun or pronoun that occurs elsewhere in a sentence. The *accessibility hierarchy* (*AH*) for relative clauses is given by Keenan and Comrie (1977) and illustrated in Figure 15.

Noun phrases (NP) occurring to the left of ">" are more accessible than those on the right. SU is short for subject, DO for direct object, IO for indirect object, OBL for major oblique case NP, GEN for genitive NP, OCOMP for object of comparison. The properties of the accessible hierarchy are contained in two sets of constraints.

The accessible hierarchy constraints (*AHCs*) are:

AHC1) A language must be able to relativize subjects.

ACH2) Any RC forming strategy must apply to a continuous segment of the AH.

ACH3) Strategies that apply at one point of the AH may in principle cease to apply at any lower point.

The primary relativization constraints (*PRCs*) are

PRC1) A language must have a primary RC-forming strategy.

PRC2) If a primary strategy in a given language can apply to a low position on the AH, then it can apply to all higher positions.

PRC3) A primary strategy may cut off at any point on the AH.

For a given language a deployment that can be used to relativize a clause at a specified place on the AH can also be used to relativize all more accessible clauses. The type of relativization varies from language to language. There appears to be nothing known on how the skill to deploy a relativization develops in an individual. One would expect that when a given method is applied the less accessible would take longer to process. There seems to be no psycholinguistic tests done to see if this is indeed the case.

7.2. The Berlin-Kay Universal Colour Partial Ordering

The perception of colour often involves the deployment of a colour name strategy. The effect of this is to alter the way the colour is perceived. The five principles of colour perception are:

CP1) The communicability of a referent in an array and for a particular community is very closely related to the memorability of that referent in the same array and for members of the same community.

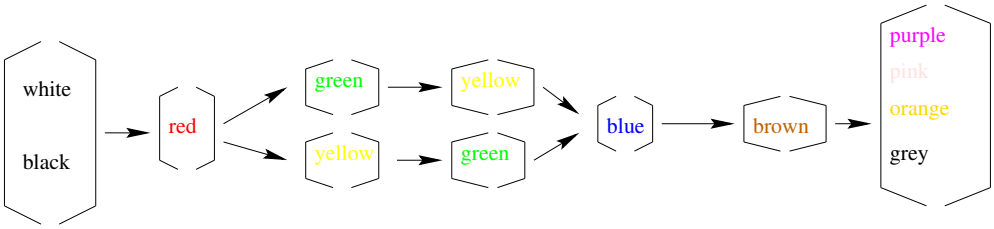


Figure 16: The Berlin-Kay Universal Colour Partial Ordering

CP2) In the total domain of colour there are eleven small focal areas in which are found the best instances of the colour categories named in any particular language. The focal areas are human universals, but languages differ in the number of basic colour terms they have: they vary from two to eleven.

CP3) Colour terms appear to evolve in a language according to the Berlin and Kay (1969) universal partial ordering illustrated by Figure 16.

CP4) Focal colours are more memorable and easier to recognize than any other colours, whether or not the subject speaks a language having a name for the colour.

CP5) The structure of the colour space determined by multi-dimensional scaling of perceptual data is probably the same for all human communities and it is unrelated to the space yielded by naming data.

Again there is a *culturally determined linguistic partial ordering* (or hierarchy). On this occasion it determines the semantic content of individual words rather than syntax rules. Again there appears to be nothing known on how the skill develops in an individual, or any timing tests on the possession of a colour name strategy. The existence of two separate hierarchical *partial orderings* suggests that there is a general mechanism for there construction. Most members of a community seem to develop these culturally determined skills suggesting that the capacity to develop them is usually innate but their manifestation depends on environment.

Bibliography

Andreas Dress, Vincent Moulton and Werner Terhalle. T-theory: An overview. *Europ. J. Combinatorics*, 17:161–175, 1996.

B. Dragovich, A.Yu. Khrennikov, S.V. Kozyrev and I.V. Volovich. On p-adic mathematical physics, p-adic numbers. *Ultrametric Analysis, and Applications Journal*, 1:1–90, 2009.

Backofen, Rogers and Vijay-Shankar. A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language and Information*, 4(1):5–39, 1995.

Baldi, P. and E.B. Baum. Bounds on the size of ultrametric structures. *Phys.Rev.Lett.*, 56:182–184, 1986.

Berlin, Brent and Paul Kay. *Basic Colour Terms*. University of California Press, 1969.

Bjorken, James D. and Sidney D. Drell. *Relativistic Quantum Fields*. McGraw Hill, 1965.

Boettcher, S. and M. Paczuski. Aging in a model of self-organized criticality. *Phys.Rev.Lett.*, 79: 889, 1997.

Botha, Rudolf P. *Challenging Chomsky: The Generative Garden game*. Blackwell Oxford, 1965.

Brekke, L. and Peter G.O. Freund. P-adelic numbers in physics. *Phys.Rep.*, 233:1–166, 1993.

Chomsky, Noam. *Knowledge of Language, Its Nature, Origin and Use*. Praeger Publishers, New York, 1986a.

Chomsky, Noam. *Barriers*. MIT Press,Cambridge, MA, 1986b.

Christiansen, Henning. Chr as grammar formalism. 2001.

Cowan, Nelson. The magical number 4 in short-term memory:a reconsideration of mental storage capacity. *Behavioral & Brain Sciences*, 24(1), 2001.

Delon, F. Espaces ultramétriques. *J. Symbolic Logic*, 49:405, 1984.

Evans, Nicholas and Stephen Levison. The myth of language universals: Language diversity and its importance for cognitive science. *Behavioural and Brain Sciences*, 32(05):429–448, 2009.

Frazier, Lyn. Sentence processing: A tutorial review. *Attention & Performance*, XII:559–586, 1987.

Guénoche, A. Order distance associated with hierarchy. *J.Classification*, 14:101, 1997.

Haegeman, Liliane. *Introduction to Government and Binding Theory*. Blackwell, Oxford, 1994.

Hammer, Hanno. Tree structure, entropy, and the action principle for neighbourhood topologies. Technical report, Cambridge, DAMTP, 1998.

Higgs, P.G. Overlaps between rna secondary structure. *Phys.Rev.Lett.*, 76:704–707, 1996.

Jackendoff, R.S. *X Syntax. A Study of Phrase Structure*. MIT Press, Cambridge, Mass, 1977.

Jardine, N. and R. Sibson. *Mathematical Taxonomy*. John Wiley and Sons, 1971.

Johnson, N.F. The psychological reality of phrase-structure rules. *Journal of Verbal Learning and Verbal Behaviour*, 4:469–475, 1965.

Karwowski, W. and R.V. Mendes. Hierarchical structures and asymmetric stochastic processes on p-adics and adeles. *J.Math.Phys.*, 35:4637, 1994.

Kayne, Richard S. Unambiguous paths. *Levels of Syntactic Representation*, 5, 1981.

Keenan, E.L. and Bernard Comrie. Noun phrase accessibility and universal grammar. *Linguistic Inquiry*, 8:63–99, 1977.

Levelt, W.J.M. Hierarchical clustering algorithms in the psychology of grammar. *Advances in psycholinguistics*, 1970.

Lockward, D.G. *Introduction to Stratification Grammar*. Harcourt Brace Jovanovich Inc.,New York., 1972.

McCloskey, J. *Syntatic Theory*. Cambridge University Press, Cambridge, 1988.

Miller, George A. The magical number seven, plus or minus two:some limits on our capacity to process information. *Psy.Rev.*, 63:81–97, 1956.

Murtagh, F. On ultrametricity, data coding, and computation. *Journal of Classification*, 21:167–184, 2004.

Ogielchi, A.T. and D.L. Stein. Dynamics on ultrametric spaces. *Phys.Rev.Lett.*, 55:1634–1637, 1985.

Parga, N. and M.A. Virasoro. The ultrametric organization of neural net memories. *J.de Physique*, 47:1857, 1986.

Perlman, E.M. and M. Schwarz. The directed polymer problem. *Europhys.Lett.*, page 227, 1992.

Prince, A. and P. Smolensky. Optimality: From neural networks to universal grammar. *Science*, page 1604, 1997.

R. Rammal, G. Toulouse and M.A. Virasoro. Ultrametricity for physicists. *Rev.Mod.Phys.*, 58: 765–788, 1986.

Rissanen, Jorma. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11:416–431, 1983.

Roberts, Mark D. Name strategy: Its existence and implications. *Int.J.Computational Cognition*, 3:1–14, 2005.

Rutten, J.J.M.M. Elements of generalized ultrametric domain theory. *Theor.Comp.Sci.*, 170:349, 1996.

Schweinberger, M. and T.A.B. Snijders. Settings in social networks: A measurement model. *Sociological Methodology*, 23:307–341, 2003.

Shepard, R.N. and P. Arabie. Additive clustering: Representative of similarities as combinations of discrete overlapping properties. *Psychological Review*, 86:87–123, 1979.

Sneath, P.H. and R.R. Sokal. *The Principles and Practice of Numerical Classification*. W.H.Freeman and Company, San Francisco., 1973.

Sorton, George. Introduction to the history of science. *Introduction to the History of Science*, III: 552, 1947.

Vlad, M.O. Fractal time, ultrametric topology and fast relation. *Phys.Lett.*, A189:299–303, 1994.

Weissman, M.B. What is spin glass. *Rev.Mod.Phys.*, 65:829, 1993.

Young, M.R. and W.S. DeSarbo. A parametric procedure for ultrametric tree estimation from conditional rank order proximity data. *Psychometrika*, 60:47, 1995.

Zadrozny, Wlodek. Minimum description length and compositionality. *Computing Meaning*, 1: 113–128, 1999.

Address for correspondence:
Mark D. Roberts
robemark@gmail
Flat 44, The Cloisters, 83 London Road,
Guildford, GU1 1FY, United Kingdom

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015 131–138

Exact Expected Average Precision of the Random Baseline for System Evaluation

Yves Bestgen

CECL, Université catholique de Louvain

Abstract

Average precision (AP) is one of the most widely used metrics in information retrieval and natural language processing research. It is usually thought that the expected AP of a system that ranks documents randomly is equal to the proportion of relevant documents in the collection. This paper shows that this value is only approximate, and provides a procedure for efficiently computing the exact value. An analysis of the difference between the approximate and the exact value shows that the discrepancy is large when the collection contains few documents, but becomes very small when it contains at least 600 documents.

1. Introduction

Many tasks in information retrieval, computational linguistics and machine learning aim at finding relevant items among a collection of items, such as documents matching a query, subjective statements, collocations, semantic neighbors, sentences between which textual entailment holds, and so forth. To evaluate the proposed systems, precision (the proportion of retrieved documents that are relevant) and recall (the proportion of relevant documents that have been retrieved) are favored. When the system ranks the documents according to their estimated relevance, performance is typically assessed through a precision-recall curve that is summarized by average precision (AP) (Büttcher et al., 2010; Robertson, 2008). AP is equal to the average of the precision value obtained after each relevant document is retrieved (i.e., when recall increases) and corresponds to the area under the uninterpolated precision-recall

 Corresponding author: yves.bestgen@uclouvain.be
Cite as: Yves Bestgen. Exact Expected Average Precision of the Random Baseline for System Evaluation. The Prague Bulletin of Mathematical Linguistics No. 103, 2015, pp. 131–138. doi: 10.1515/pralin-2015-0007.

curve (PR) (Voorhees and Harman, 1999). More formally,

$$AP = \frac{\sum_{i=1}^{N} \frac{i}{n}}{R} \quad (1)$$

where R is number of relevant documents and n the rank of the ith document according to the system. This rank goes from 1 to N, the number of documents in the collection (see Robertson (2008, p. 689) for another, yet equivalent, formula of AP).

If AP is often used to compare the performance of different systems on the same test collection, with each system serving as benchmark for the others, the AP obtained by a system is sometimes compared to a *random baseline* AP: the expected AP that would be obtained by a system that ranks the documents in a completely random way (e.g. Marszalek et al., 2009; Nakano et al., 2011; Pecina, 2010; Pohlmeyer et al., 2011; Ramisch et al., 2008; Rasiwasia et al., 2010). This baseline AP is considered to be equal to the proportion of relevant documents in the collection, also called the category prevalence. This paper shows that this value is only approximate (section 2), and provides a procedure for efficiently computing the exact value (section 3). An analysis of the difference between the approximate and the exact value shows that the discrepancy is large when the collection contains few documents, but becomes very small when it contains at least 600 documents (section 4).

2. The Proportion of Relevant Documents is not Equal to the Expected AP for the Random Baseline

Researchers employing the proportion of relevant documents as the expected value of the random baseline AP do not justify the choice of this value. Presumably, they start from the fact that AP is equal to the area under the PR curve, and that a system that ranks the documents in a completely random way should uniformly distribute the relevant documents along the ranking. The proportion of relevant documents retrieved relative to the total number of documents considered should thus be constant at all ranking positions. It follows that the corresponding PR curve is a straight line whose intercept is the proportion of relevant documents in the collection ($p = R/N$) and whose slope is 0. The area under this "curve" is the proportion of relevant documents.

A very simple example is sufficient to show that the proportion of relevant documents is only an approximation of the actual AP for the random baseline. Consider a test collection consisting of five documents, of which two are relevant: p is thus 0.40. It is very easy to list all the possible rankings and to compute their AP as shown in Table 1. In this table, document relevance is represented by a binary variable set to one when the document is relevant. Since all these rankings are equally probable for a system that ranks documents randomly, the expected AP of the random baseline is the mean AP computed on all possible permutations. For this example, it is thus not 0.40 (R/N), but 0.593.

Ranking					AP
1st	2nd	3rd	4th	5th	
1	1	0	0	0	(1/1 + 2/2) / 2 = 1.00
1	0	1	0	0	(1/1 + 2/3) / 2 = 0.83
1	0	0	1	0	(1/1 + 2/4) / 2 = 0.75
1	0	0	0	1	(1/1 + 2/5) / 2 = 0.70
0	1	1	0	0	(1/2 + 2/3) / 2 = 0.58
0	1	0	1	0	(1/2 + 2/4) / 2 = 0.50
0	1	0	0	1	(1/2 + 2/5) / 2 = 0.45
0	0	1	1	0	(1/3 + 2/4) / 2 = 0.42
0	0	1	0	1	(1/3 + 2/5) / 2 = 0.37
0	0	0	1	1	(1/4 + 2/5) / 2 = 0.33
					Sum = 5.93
					Expected AP = 5.93 / 10 = 0.593

Table 1. Expected AP of the random baseline for $N = 5$ *and* $R = 2$

3. An Accurate and Efficient Procedure for Calculating the AP for the Random Baseline

To compute the expected AP of the random baseline for other values of N and R (or p), one could imagine using the procedure outlined in Table 1. The problem is that it would require enumerating a very large number of permutations when N is large and R is not too close to 0 or to N. It corresponds to the number of different permutations of N objects when some of these are identical, that is, $N!/(R! \times (N - R)!)$. This results in more than 17,000 billion different rankings to list for $N = 100$ and $p = 0.10$.

Looking at this table, a much more efficient solution can be proposed. If the two divisors (R and the total number of different permutations) are set aside, there remains a sum of precision scores at rank n (i.e., i/n), n corresponding to the possible positions in the ranking of each ith relevant document. For each value of i (i ranging from 1 to R), there are $N - R + 1$ possible ranks, since the ith relevant document cannot occur before the ith rank or after the $N-R+i$ rank; otherwise, there are not enough positions available for the $R - i$ remaining relevant documents. Furthermore, for each value of i, there are in theory a total of $N!/(R! \times (N - R)!)$ precision scores to compute, since the ith relevant document is present in every possible permutation. But the problem can be reformulated in terms of the probability that the ith relevant document occurs at each of the $N - R + 1$ possible ranks, all the probabilities computed for a given i summing to 1. This formulation requires the calculation of only $R \times (R+N-1)$ values.

i	n	dhyper(i,N,N-R,n)	i/n	Final prob.	P@n	Contribution to AP
1	1	0.4	1.00	0.4	1.00	0.400
1	2	0.6	0.50	0.3	0.50	0.150
1	3	0.6	0.33	0.2	0.33	0.067
1	4	0.4	0.25	0.1	0.25	0.025
2	2	0.1	1.00	0.1	1.00	0.100
2	3	0.3	0.67	0.2	0.67	0.134
2	4	0.6	0.50	0.3	0.50	0.150
2	5	1.0	0.40	0.4	0.40	0.160
						Sum = 1.186
						Expected AP = 1.186 / 2 = 0.593

Note: *dhyper()* returns the density for the hypergeometric function. *P@n* stands for *precision at rank n*.

Table 2. Calculation of the expected AP of the random baseline for $N = 5$ *and* $R = 2$

The proposed procedure is, therefore, to calculate, for each rank n, the probability that the ith relevant document occurs at that rank, producing a precision score at rank n equal to i/n. This probability is equal to the probability of having i successes in n draws *without replacement* from a population of size N containing R successes and $N - R$ failures, with the additional condition that the ith success occurs at the last draw (i.e., at rank n). The first part of this probability is given by the hypergeometric distribution whose formula is:

$$P(X = i) = \frac{\binom{R}{i}\binom{N-R}{n-i}}{\binom{N}{n}} \tag{2}$$

where $\binom{R}{i}$ is a binomial coefficient, corresponding here to $i!/(R!(R-i)!)$. Regarding the additional condition, the probability of a success at the last draw when there are i successes in n draws is obviously i/n. Multiplying this final probability by the precision score at rank n produces the contribution of each ith relevant document to the total sum of AP, and it only remains to divide this sum by R to obtain the expected AP of the random baseline.

Table 2 applies this calculation procedure[1] to the example of Table 1. In this example, the gain in number of operations is very small (eight instead of 10), but for $N = 100$ and $R = 10$, it is reduced from more than 17,000 billion to 910. Calculating

[1] In this table, the *Final prob.* values for each i sum to 1, as explained. This is not the case for the probabilities from the hypergeometric distribution alone, which sum to 1 for a given n only if one adds the probabilities for all possible i (i.e., the number of successes) including 0 success.

a hypergeometric probability takes more time than finding a possible permutation, but extremely efficient procedures for calculating these probabilities are available in every major statistical software.

The very simple R function (R Core Team, 2013) given below implements the calculation procedure of the expected AP of the random baseline for any values of N and R.

```
RandomAPExact = function(N=0, R=0) {
  ap = 0
  for (i in 1:R) {
   for (n in i:(N-R+i)) {
    ap = ap + dhyper(i,R,N-R,n)*(i/n)*(i/n)
   }
  }
  ap = ap/R
  ap
 }
Function call: RandomAPExact(10,4)
Result: [1] 0.5285979
```

For N = 100 and p = 0.10, this function takes 0.011 seconds to compute the solution, and just over 131 seconds for N = 10000 and p = 0.40 on an *Intel Core i5* 2.66GHz processor. If there is no doubt that the exact procedure is computationally intensive compared to the calculation of the approximate value, the R code provided allows to calculate it very easily and it should only be used once for an evaluation task. What is a handful of minutes compared to the time required for the development of an IR system and for its evaluation?

4. How Large is the Difference Between the Exact Value and the Usual Estimate?

To get an idea of the importance of differences between the exact AP and the approximate AP for the random baseline, Figure 1 shows the evolution of this difference for many values of N and p, the exact value being systematically larger than the approximate value. As can be seen, the difference decreases when N increases or p increases. As soon as N is at least equal to 600, it is less than 0.01 for all tested values of p.

5. Conclusion

This paper shows that the AP for the random baseline usually used in information retrieval and computational linguistics is only an approximation of the exact AP, and it presents an efficient procedure to compute the latter. An analysis of the dif-

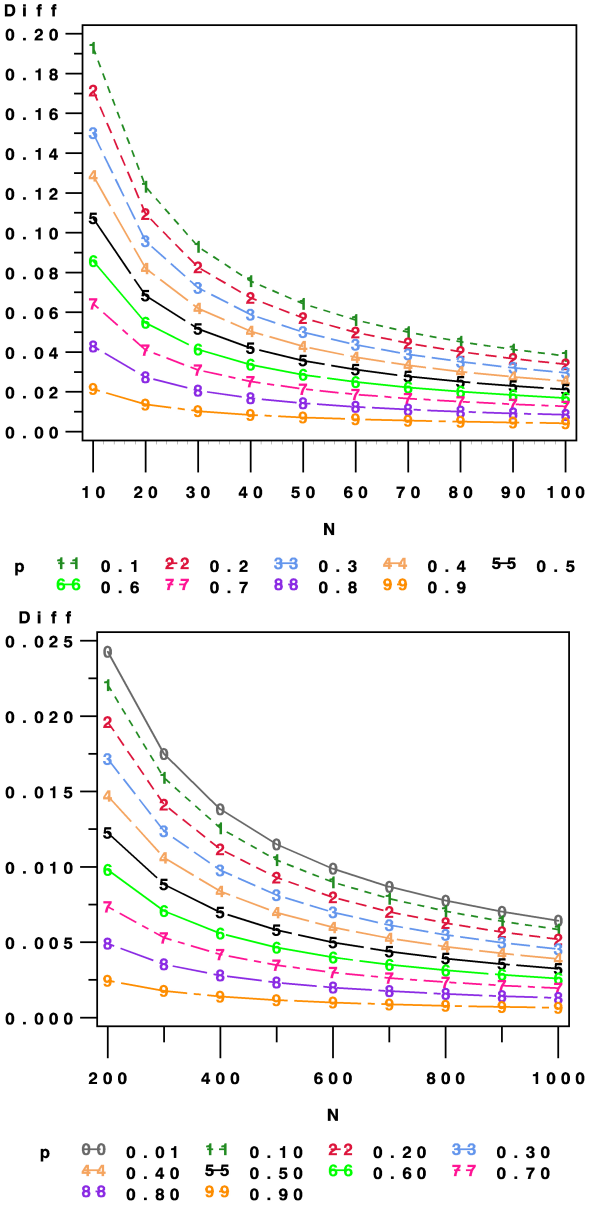


Figure 1. Difference between the exact and the approximate AP of the random baseline for several values of N *and* p

ference between the exact value and the approximate value shows that the discrepancy between them reduces when the size of the collection of documents increases. While many evaluations of IR systems are performed on very large collections of documents, some research areas use much smaller collections because of the difficulties encountered in their constitution (i.e., less resourced languages, emerging tasks or tasks requiring complex relevance judgment that can only be performed by human experts). The smallness of the collections can be further enhanced by the use of a random under-sampling procedure advocated by Jeni et al. (2013) to reduce the impact of large imbalance between the positive and negative examples on performance metrics.

In conclusion, it can be recommended that researchers who plan to compare their system to the random baseline AP use the proposed procedure to calculate the exact expected value when the test collection is of limited size or, when there are at least 600 documents in the collection, state in their report that the proportion of relevant documents in the collection is an excellent approximation of the exact value.

Acknowledgements

Yves Bestgen is a Research Associate with the Belgian Fund for Scientific Research (F.R.S-FNRS).

Bibliography

Büttcher, Stefan, Charles Clarke, and Gordon Cormack. *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press, 2010.

Jeni, László, Jeffrey Cohn, and Fernando De La Torre. Facing imbalanced data - recommendations for the use of performance metrics. In *ACII '13: 2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 245–251, 2013.

Marszalek, Marcin, Ivan Laptev, and Cordelia Schmid. Actions in context. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2929–2936, 2009.

Nakano, Takuho, Akisato Kimura, Hirokazu Kameoka, Shigeki Miyabe, Shigeki Sagayama, Nobutaka Ono, Kunio Kashino, and Takuya Nishimoto. Automatic video annotation via hierarchical topic trajectory model considering cross-modal correlations. In *Acoustics, Speech and Signal Processing*, pages 2380–2383, 2011.

Pecina, Pavel. Lexical association measures and collocation extraction. *Language Resources & Evaluation*, 44:137–158, 2010.

Pohlmeyer, Eric, Jun Wang, David Jangraw, Bin Lou, Shih-Fu Chang, and Paul Sajda. Closing the loop in cortically-coupled computer vision: a brain–computer interface for searching image databases. *Journal of Neural Engineering*, 8:1–14, 2011.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2013.

Ramisch, Carlos, Paulo Schreiner, Marco Idiart, and Aline Villavicencio. An evaluation of methods for the extraction of multiword expressions. In *LREC Workshop towards a Shared Task for Multiword Expressions*, pages 50–53, 2008.

Rasiwasia, Nikhil, Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Gert Lanckriet, Roger Levy, and Nuno Vasconcelos. A new approach to cross-modal multimedia retrieval. In *ACM MM '10, the International Conference on Multimedia*, pages 251–260, 2010.

Robertson, Stephen. A new interpretation of average precision. In *31st Annual international ACM SIGIR Conference*, pages 689–690, 2008.

Voorhees, Ellen and Donna Harman. Overview of the seventh text retrieval conference. In *Seventh Text REtrieval Conference (TREC-7)*. NIST Special Publication, 1999.

Address for correspondence:
Yves Bestgen
yves.bestgen@uclouvain.be
Centre for English Corpus Linguistics
Université catholique de Louvain
Place du cardinal Mercier, 10
Louvain-la-Neuve, 1348, Belgium

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015 139–159

A Python-based Interface for Wide Coverage Lexicalized Tree-adjoining Grammars

Ziqi Wang, Haotian Zhang, Anoop Sarkar

School of Computing Science, Simon Fraser University

Abstract

This paper describes the design and implementation of a Python-based interface for wide coverage Lexicalized Tree-adjoining Grammars. The grammars are part of the XTAG Grammar project at the University of Pennsylvania, which were hand-written and semi-automatically curated to parse real-world corpora. We provide an interface to the wide coverage English and Korean XTAG grammars. Each XTAG grammar is lexicalized, which means at least one word selects a tree fragment (called an *elementary tree* or *etree*). Derivations for sentences are built by combining etrees using substitution (replacement of a tree node with an etree at the frontier of another etree) and adjunction (replacement of an internal tree node in an etree by another etree). Each etree is associated with a feature structure representing constraints on substitution and adjunction. Feature structures are combined using unification during the combination of etrees. We plan to integrate our toolkit for XTAG grammars into the Python-based Natural Language Toolkit (NLTK: nltk.org). We have provided an API capable of searching the lexicalized etrees for a given word or multiple words, searching for a etree by name or function, display the lexicalized etrees to the user using a graphical view, display the feature structure associated with each tree node in an etree, hide or highlight features based on a regular expression, and browsing the entire tree database for each XTAG grammar.

1. Introduction

Tree-Adjoining Grammars (TAG) represents a tree manipulating system. TAG was first proposed in (Joshi et al., 1975) as grammatical formalism that extends context-free grammars (CFG) with an extended domain of locality (namely a tree structure

 Corresponding author: anoop@sfu.ca
Cite as: Ziqi Wang, Haotian Zhang, Anoop Sarkar. A Python-based Interface for Wide Coverage Lexicalized Tree-adjoining Grammars. The Prague Bulletin of Mathematical Linguistics No. 103, 2015, pp. 139–159.
doi: 10.1515/pralin-2015-0008.

rather than a CFG rule). The main reasons to consider TAG as a suitable formalism for the analysis of the syntax of natural language are:

- TAG can be viewed as a system that can be used for lexicalization of a CFG. Unlike the Griebach normal form for CFGs, a lexicalized TAG can preserve the tree set of the original CFG (Schabes et al., 1988; Schabes and Waters, 1995). Most linguistic applications of TAG use a lexicalized TAG (LTAG).
- Like CFGs, TAGs are parsable in polynomial time.
- TAG can generate crossing dependencies which are widespread in natural language, while CFGs can only generate nested dependencies.

A comprehensive introduction to TAG, both the formalism and the linguistic application of TAG appears in (Joshi and Schabes, 1997). Large scale LTAG grammars have been constructed by hand for English as part of the XTAG project at the University of Pennsylvania (UPenn) (XTAG Group, 2001) and French (at the TALANA group, University of Paris 7, France) and somewhat smaller ones for German (at DFKI, Saarbrücken, Germany), Korean (at UPenn), Chinese (at UPenn), and Hindi (at CDAC, Pune, India). The earliest stochastic variant of TAG was proposed by (Resnik, 1992; Schabes, 1992). LTAG grammars have been extracted from annotated corpora (Xia et al., 2001; Xia, 2001; Chiang, 2000; Chen and Vijay-Shanker, 2000), which in turn have been used for lexicalized statistical parsing (Chiang, 2000; Sarkar, 2001). In this paper, our focus will be on the hand-written large scale LTAG grammars developed as part of the XTAG project at the University of Pennsylvania (XTAG Group, 2001).

The goal of this work is to provide a modern programming interface to the large scale LTAG grammars developed as part of the XTAG project at the University of Pennsylvania. We provide a conversion of the grammar files for English and Korean wide coverage LTAG grammars. To this end, we describe in this paper our Python-based API for viewing and manipulating these LTAG grammars. We plan to incorporate our modules into the Python-based Natural Language Toolkit (NLTK) project so that anybody who downloads the NLTK project can have access to the linguistic annotations that are part of the LTAG grammars developed during the XTAG project. All the data is also available through online repositories and packaged for distribution and convenient use by our codebase. This is particularly useful as the software that was originally developed for manipulating these grammars by the XTAG project is no longer being maintained.

2. Tree-adjoining grammars

In this section we summarize the representation used in the XTAG grammars. The material here is a summary of the extended description provided in the XTAG technical report describing the English grammar (XTAG Group, 2001).

Tree-adjoining grammar (TAG) is a formal tree rewriting system. TAG and Lexicalized Tree-Adjoining Grammar (LTAG) have been extensively studied both with

respect to their formal properties and to their linguistic relevance. TAG and LTAG are formally equivalent, however, from the linguistic perspective LTAG is the system we will be concerned with in this paper. We will often use these terms TAG and LTAG interchangeably.

The motivations for the study of LTAG are both linguistic and formal. The elementary objects manipulated by LTAG are structured objects (trees or directed acyclic graphs) and not strings. Using structured objects as the elementary objects of the formal system, it is possible to construct formalisms whose properties relate directly to the study of strong generative capacity (i.e., structural descriptions), which is more relevant to the linguistic descriptions than the weak generative capacity (sets of strings).

Rather than giving formal definitions for LTAG and derivations in LTAG we will give a simple example to illustrate some key aspects of LTAG. We show some elementary trees of a toy LTAG grammar of English. Figure 1 shows two elementary trees for a verb such as *likes*. The tree α_1 is anchored on *likes* and encapsulates the two arguments of the verb. The tree α_2 corresponds to the object extraction construction. Since we need to encapsulate all the arguments of the verb in each elementary tree for *likes*, for the object extraction construction, for example, we need to make the elementary tree associated with *likes* large enough so that the extracted argument is in the same elementary domain. Thus, in principle, for each 'minimal' construction in which *likes* can appear (for example, subject extraction, topicalization, subject relative, object relative, passive, etc.) there will be an elementary tree associated with that construction. By 'minimal' we mean when all recursion has been factored away. This factoring of recursion away from the domain over which the dependencies have to be specified is a crucial aspect of LTAGs as they are used in linguistic descriptions. This factoring allows all dependencies to be localized in the elementary domains. In this sense, there will, therefore, be no long distance dependencies as such. They will all be local and will become long distance on account of the composition operations, especially adjoining.

Figure 2 shows some additional trees. Trees α_3, α_4, and α_5 are initial trees and trees β_1 and β_2 are auxiliary trees with foot nodes marked with *. A derivation using the trees in Figure 1 and Figure 2 is shown in Figure 3. The trees for *who* and *Harry* are substituted in the tree for *likes* at the respective *NP* nodes, the tree for *Bill* is substituted in the tree for *think* at the *NP* node, the tree for *does* is adjoined to the root node of the tree for *think* tree (adjoining at the root node is a special case of adjoining), and finally the derived auxiliary tree (after adjoining β_2 to β_1) is adjoined to the indicated interior S node of the tree α_2. This derivation results in the **derived tree** for *who does Bill think Harry likes* as shown in Figure 4. Note that the dependency between *who* and the complement *NP* in α_2 (local to that tree) has been stretched in the derived tree in Figure 4. This tree is the conventional tree associated with the sentence.

However, in LTAG there is also a derivation tree, the tree that records the history of composition of the elementary trees associated with the lexical items in the sentence. This derivation tree is shown in Figure 5. The nodes of the tree are labeled by the

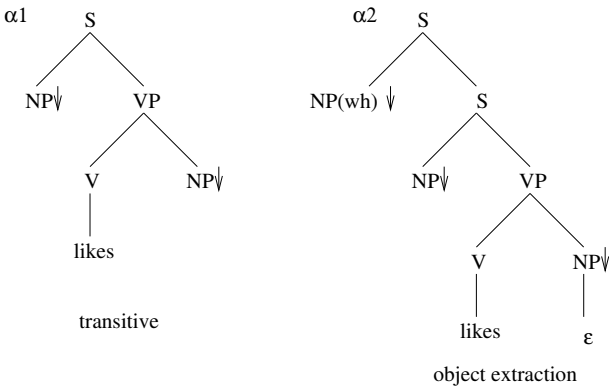


Figure 1. LTAG: Elementary trees for likes. The same predicate-argument structure can be syntactically realized in different ways. All the different syntactic transformations produce a set of elementary trees for each predicate. This set is grouped together into a tree family *for each type of predicate.*

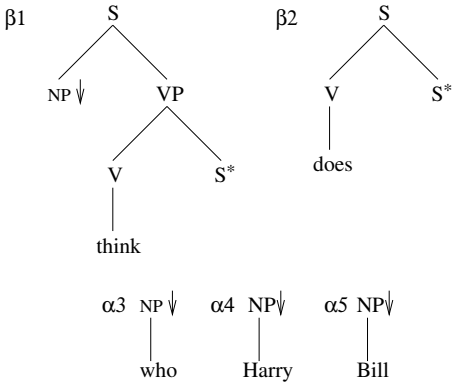


Figure 2. LTAG: Sample elementary trees

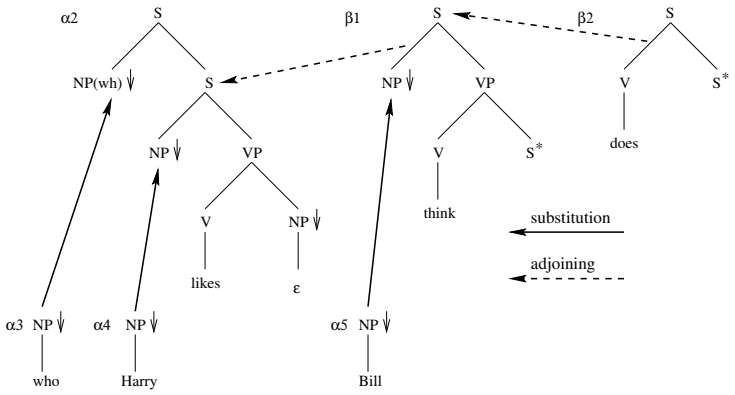


Figure 3. LTAG derivation for who does Bill think Harry likes

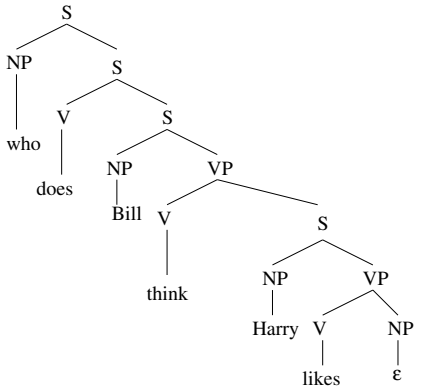


Figure 4. LTAG derived tree for who does Bill think Harry likes

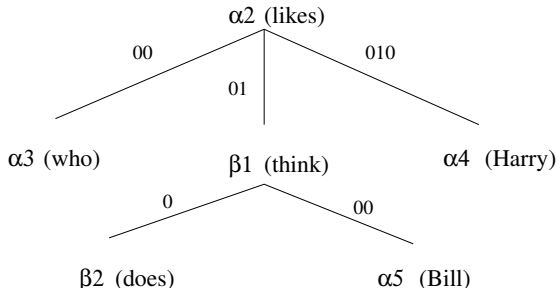


Figure 5. LTAG derivation tree

tree labels such as α_2 together with the lexical anchor.[1] The derivation tree is the primary derivation structure for LTAG: the derived tree can be directly created using the information in the derivation tree.

Large scale wide coverage grammars have been built using LTAG, the XTAG English grammar (an LTAG grammar and lexicon for English) being the largest so far (for further details see (XTAG Group, 2001).

2.1. Feature Structures and TAG

A feature structure is a set of key-value pairs (i.e. features), in which the value could be of basic type or other feature structure. Two or more keys could share the same value, and value could be shared among keys. This property facilitates interactions between tree nodes. The main operation defined for feature structure is unification, through which two feature structures are merged into one. Unification could fail if there is conflicting feature. A feature structure based TAG is a TAG with feature structures attached to tree nodes.

In the XTAG grammars, each node in each LTAG tree is decorated with two feature structures (top and bottom feature structures), in contrast to the CFG based feature structure grammars. This is necessary because adjoining can augment a tree internally, while in a CFG based grammar or even in a tree substitution grammar a tree can be augmented only at the frontier. It is possible to define adjoining and substitution (as it is done in the XTAG system) in terms of appropriate unifications of the top and bottom feature structures.

When doing substitution, the top feature of the new node is the result of unification of the root node and substitution node, while the bottom feature of the new node is

[1]The derivation trees of LTAG have a close relationship to the notion of dependency trees, although there are some crucial differences; however, the semantic dependencies are the same. See (Rambow and Joshi, 1995) for more details.

the bottom feature of the root node in substituting tree. See Figure 6 for a schematic view of how unification is done during the substitution operation in LTAG.

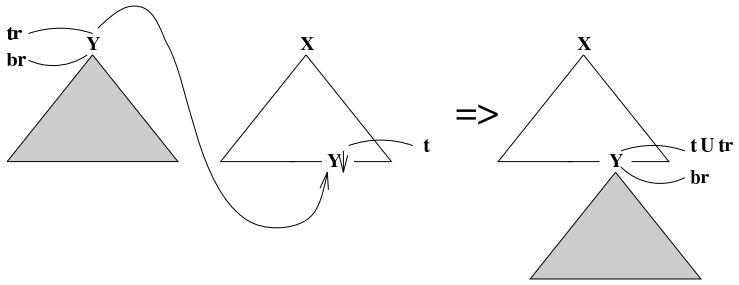


Figure 6. Substitution in feature-based LTAG

Adjunction is more complicated: after the adjunction node has been split into two, the top feature is now attached with the top half, and the bottom feature is attached with another half. In addition, in the resulting tree produced by adjunction, the top feature attached with the root of the auxiliary tree is then unified with the top feature of the top half, and similarly, the bottom feature attached with the adjunction node of the auxiliary tree is unified with the bottom feature of another half. See Figure 7 for a schematic view of how unification is done during the adjunction operation in LTAG.

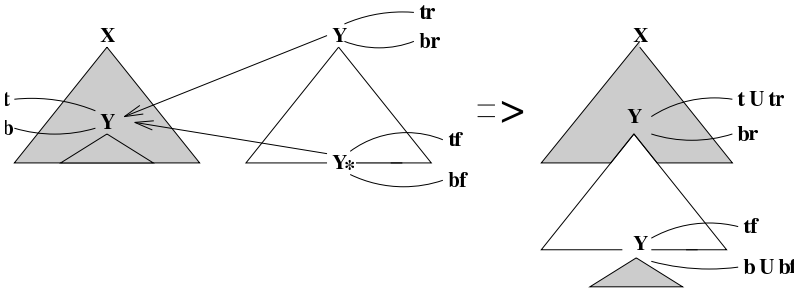


Figure 7. Adjunction in feature-based LTAG

Constraints on substitution and adjoining are modelled via these feature structures (Vijay-Shanker, 1987) (except for the null adjunction constraint or NA constraint which rules out adjunction at a node).

3. Structure of the XTAG Grammar

Each XTAG grammar (both the English and Korean grammar) is a large lexicalized Tree Adjoining Grammar. The grammar itself is represented in machine readable format in a series of files, each of which represents a different module in the grammar. In this section, we describe each module.

3.1. Trees and Tree Families

The first important concept in the XTAG grammar is the notion of a tree family. As seen in Figure 1 each predicate lexicalizes many different elementary trees where the predicate-argument structure is the same but the syntactic construction is different. All of the different syntactic realizations of a predicate are grouped into a *tree family*. In XTAG, the name of the tree family encapsulates the predicate-argument structure. For instance, the tree family name `Tnx0Vplnx1` refers to all the syntactic variations of a single predicate argument structure where `nx0` and `nx1` represent the subject and object NP arguments of the predicate respectively. The predicate is a verb represented by the `V` in the name and `pl` stands for particle. Thus, `Tnx0Vplnx1` is a tree family associated with a multi-word predicate such as *walk off*. All the trees in the tree family `Tnx0Vplnx1` are lexicalized by the anchor words *walk off* including trees for passives, wh-movement, relative clauses, etc.

Function words or non-predicates are assigned trees as well, and so there are *tree files* as well in the XTAG grammar where words lexicalize the individual trees in these tree files rather than all the trees at once.

In tree files, trees are defined as objects, encapsulating the skeleton of elementary tree, tree name, node information, feature structure for nodes, constraints, comments, and other GUI-related options, such as switches for drawing.

Tree skeletons are the frameworks of un-lexicalized elementary objects, and they are essentially DAGs (directed acyclic graphs), with tree nodes being graph nodes, and internal connections being edges. Tree nodes are again objects with at least three members: node label, node type, and node constraint. Node label is the symbol it represents, which must be a non-terminal, and appropriate measures must be taken to avoid naming collision (see below). Node type annotates whether a node on tree frontier is a substitution node, adjunction node (for internal nodes this field is empty) or head node (where the lexical item, i.e. anchor, is attached). We follow the convention that for substitution node there is a down arrow, for adjunction node there is an asterisk, and for head node, there is a diamond when displaying trees graphically. Node constraints rules out combinations that are not qualified, and should be blank if no constraint is applicable.

In order to prevent node label collision (two nodes on the same tree having identical label), which can happen in elementary trees, the notion of node label is extended to have two parts. The first part is exactly the non-terminal symbol it represents,

and the second part is appended to differentiate this node from others. Any naming scheme that guarantees uniqueness is acceptable, however, according to our convention, root nodes are given suffix `.r`, and foot nodes are given suffix `.f`. Besides, all other nodes that have a common non-terminal symbol are given an increasing numeric suffix. For example, if we have four NP nodes in an auxiliary tree, one of them being the root node, another being the foot node, and the rest two are generic nodes, then their label should be `NP.r`, `NP.f`, `NP`, `NP.1` respectively.

There are two types of elementary tree, initial tree and auxiliary tree, and in order to distinguish them, trees with a name that starts with an "alpha" α denote initial trees, and those whose name starts with a "beta" β denote auxiliary trees. This is just a naming convention.

3.2. Feature structures

XTAG is organized such that feature structures are specified in three different components of the grammar: a *Tree* database defines feature structures attached to tree *families*; a *Syn* database defines feature structures attached to lexically anchored trees; and a *Morph* database defines feature structures attached to (possibly inflected) lexical entries.

As an example, consider the verb *seems* This verb can anchor several trees, among which are trees of auxiliary verbs, such as the tree βVvx, depicted in figure 8. This tree, which is common to all auxiliary verbs, is associated with the feature structure descriptions listed in Figure 8 (independently of the word that happens to anchor it).[2]

When the tree βVvx is anchored by *seems*, the lexicon specifies additional constraints on the feature structures in this tree:

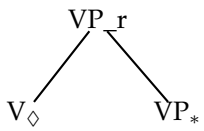
```
seem betaVvx VP.b:<mode>=inf/nom,
             V.b:<mainv> = +
```

Finally, since *seems* is an inflected form, the morphological database specifies more constraints on the node that this word instantiates, as shown in figure 9.

The full set of feature structures that are associated with the lexicalized tree anchored by *seems* is the unification of these three sets of path equations.

The atomic values in feature structures fall into three categories. The first two are generic strings and booleans (+ or -). A third type of disjunction is also used, usually represented as `a/b/c`, which means "a or b or c" (it represents a disjunctive relation).

[2]We use "handles" such as `V.b` or `NP.t` to refer to the feature structures being specified. Each node in a tree is associated with two feature structures, 'top' (.t) and 'bottom' (.b). Angular brackets delimit feature paths, and slashes denote disjunctive (atomic) values.




```
V.t:<agr> = VP_r.b:<agr>
V.t:<assign-case> = VP_r.b:<assign-case>
V.t:<assign-comp> = VP_r.b:<assign-comp>
V.t:<displ-const set1> = VP_r.b:<displ-const set1>
V.t:<mainv> = VP_r.b:<mainv>
V.t:<mode> = VP_r.b:<mode>
V.t:<neg> = VP_r.b:<neg>
V.t:<tense> = VP_r.b:<tense>
VP.t:<assign-comp> = ecm
VP.t:<compar> = -
VP.t:<displ-const set1> = -
VP_r.b:<compar> = -
VP_r.b:<conditional> = VP.t:<conditional>
VP_r.b:<perfect> = VP.t:<perfect>
VP_r.b:<progressive> = VP.t:<progressive>
```

Figure 8. An example tree and its associated feature structure descriptions

```
seems   seem  V   <agr pers> = 3,
                  <agr num> = sing,
                  <agr 3rdsing> = +,
                  <mode> = ind,
                  <tense> = pres,
                  <assign-comp> = ind_nil/that/rel/if/whether,
                  <assign-case> = nom
```

Figure 9. The morphological database entry for seems

3.3. Morphology

The morphology information in XTAG establishes mappings from inflected words to a list of tuples. The morphology file contains a mapping from inflected words to the root word, part of speech (POS) and inflectional information. If there are multiple records returned, then they are organized as a list. Figure 10 contains an example of a morphology entry in XTAG. The leftmost word of the first line, *facile*, is the index of record. The other two columns are the stem form and part of speech (POS) respectively.

Inflected Word	Stem	POS	Inflectional Information
facile	facile	A	
cajoles	cajole	V	3sg PRES
chills	chill	N	3pl
chills	chill	V	3sg PRES

Figure 10. Morphology Example

Index	Sense 1	Sense 2	Sense 3	Sense 4
crabs	crab N 3pl	crabs N 3sg	crab V PPART STR	crab V 3sg PRES

Figure 11. Multiple Records Example

3.4. Syntax and Feature Templates

Syntax information in XTAG consist of two components: the syntactic mapping from stems to trees and the list of feature templates.

The syntactic database stores mappings from a stem to a list of structures. It is queried using an uninflected word as index. The query returns a list of syntactic items. Each syntactic item is recognized as a structure with header and body, which contains information to lexicalize TAG trees.

Each entry has a header that contains ENTRY and POS (part of speech) keys. ENTRY and POS must appear in pairs, the values of which are inflected word and POS tag respectively. Duplicated pairs of ENTRY and POS are allowed, and this happens when the identifier is a phrase rather than a single word. An example of the phrase "walk off" is presented below.

«INDEX»walk«ENTRY»walk«POS»V«ENTRY»off«POS»PL«FAMILY»Tnx0Vplnx1

After the header the rest of the entry contains either TREES or FAMILY and FEATURES. These three are all optional, but in order to define a non-trivial syntactic item, at least one of TREES and FAMILY must exist. TREES selects a group of trees by the file name of one or more generic tree files, while FAMILY selects a group of trees by the file name of one or more family files. Once the groups are selected, all trees in these groups will be lexicalized. In the example above, the value of FAMILY is "Tnx0Vplnx1"; therefore, all trees defined in that file will be lexicalized using "walk" and "off".

The FEATURES key in the body specifies feature structures for lexicalization. Please note that features are also defined in morphology files. The reason of scattering them in two separate files is that features in morphology file are associated with inflectional information, while those in syntactic database capture lexical idiosyncrasies. Features from both files should be applied to the trees.

@1sg	@1st, @sg, <agr 3rdsing> = -!
#_AWH+	A.b:<wh> = +!

Figure 12. Feature Template Example

In a feature template, feature identifiers from the morphology and from syntax should not be mixed up. Although they coexist in the same file, appropriate measures must be taken to help distinguish one from another. We extracted two lines from our implementation of feature template as an example.

Lines starting with a "@" is recognized as inflectional (morphological) features, while lines starting with a "#" is recognized as syntactic database features. Besides, "!" is used to terminate one line. In addition, shorthand is allowed when defining feature entries. Reference to already defined feature structures would expand them in-place as part of the definition.

3.5. Configuration files

Configuration files consist of metadata files and list of trees, families and other information about the grammar. Also included here are properties (represented as feature structures) expected for every root node in a derivation tree (for instance: the main clause should carry tense, e.g. *the toy exploded* which rules out untensed main clauses such as ∗*the toy to explode* — this requirement is implemented as a feature structure that will be unified with the root node of the elementary tree which is the root of the derivation tree for an input sentence).

4. A Client-Server Architecture

Our XTAG grammar viewer is divided into two components: the backend and frontend. The backend is responsible for data processing, such as finding TAG trees for a given word; while the frontend implements presentation and user interaction, such as drawing a TAG tree on the frame buffer with feature structures. We have designed an API that bridges the frontend and backend to make things more modular, e.g. one could use the backend API to iterate through the grammar and one could use the frontend API to draw elementary trees that are not in the XTAG grammar.

The server and client could be running on different machines, communicating using network protocols, such as remote procedure call (RPC), or even web protocol like HTTP. In the following, we use the term "backend" and "server", "frontend" and "client" interchangeably.

Interface	Description
unpack_data()	Reads XTAG dataset, and returns memory objects
dump_binary()	Dumps memory objects onto file system
load_binary()	Loads memory objects from file system

Figure 13. Data Preprocessing Interface Description

5. Backend

In this section, we present the backend of grammar viewer. As previous section has indicated, the backend serves as a data processor, and handles requests from the client.

5.1. Data Preprocessing

Data preprocessing or initialization, happens only once on startup. Data preprocessing takes the raw source grammar files and creates machine readable data structures on which the grammar viewer operates. In the reader step unpack_data() should take the path or descriptor of the configuration file, extracting global user path, relative file paths, as well as file names, and concatenate the global user path with each relative file path and file name respectively (possibly we also need to append a suffix using file type information). Then it reads contents of each file, sending them for unpacking into memory, and returns these data structures. Since there is critical information contained in the configuration file, it should be unpacked and included as well.

Developers can dump a binary image of unpacked memory objects onto the file system after loading for the first time, and only use that binary dump for later. This reduces processing time and file size. Interfaces for this purpose are defined as dump_binary() and load_binary() respectively.

5.2. Forward Searching

Forward searching, also known as tree selection (please note that we use the term "tree selection" to refer to using tree file name to include all TAG trees in that file in Section 3, "Tree Files" Section), is the process of selecting TAG trees given a word or phrase, and then to lexicalize the selected trees with information about the word or phrase. This process involves many individual steps, and could be broken down into smaller independent steps:

After receiving user input from the client, which is usually a string consisting of one or more words, the first step is to split the string into tokens, and each token is

an inflected word. These tokens are then passed to a series of procedures, including morphological analyzer, syntactic analyzer, feature manager and tree manager. Eventually the backend returns a list of TAG trees which are lexicalized by the tokens with feature structures attached to tree nodes.

The morphological analyzer accepts an inflected word as argument, and returns a set of its possible stem, POS tag, and morphological feature structures. This is accomplished by querying the morphological database. This database, as described in Section 3, is designed to be indexed using an inflected word, and the result is a list of records. Each record is a three tuple of the stem (root word, we use these two interchangeably), POS tag and feature structures. These records are then returned as a list. This procedure is defined as `word_to_morph()`.

The syntactic analyzer takes output from the morphological analyzer, and if there are multiple words (i.e. the lexical entry is a phrase), then we need to collect them using separate calls to the morphological analyzer into an aggregation, and then return the trees for the aggregated set of words. The syntactic analyzer extracts syntactic information from the syntactic database. As we have discussed in Section 4, the syntactic database is indexed using the root word from previous step, and the query returns a set of syntactic items. Each item has a header acting as an identifier, which is a stream of ENTRY and POS combinations, and a body that contains at least one of TREES, FAMILY and FEATURES. The syntactic analyzer combines morphological information with syntactic information by iterating through all morphological tuples, querying the syntactic database using the root word (if we are handling phrases, then only use the first one), and examine the result. Only those syntactic items whose identifier matches the one constructed using the root word and POS in the parameter is reserved; others are discarded, because their morphological and syntactical information do not conform. Once the syntactic items have been determined, the function returns with tree files, family files, and features extracted from those items. This procedure is defined as `morph_to_syntax()`.

The feature manager manages feature structures. This component would return a feature structure when called with a feature identifier. To avoid instantiating every feature structure objects during initialization, feature manager could be implemented with a cache that stores recently used feature structures, only duplicating and returning them when requested, and all other features remains un-instantiated until they are requested for the first time. Such a parse-on-demand manner would moderately save initialization time, while introducing slight performance harm later. The interface to access feature structures is defined as `get_feature_struct()`.

The tree manager, similar to feature manager, manages trees. However, it maintains tree database at different levels. If trees are requested by tree family name then a list of trees in that tree family is returned. The request could also be for all trees in a file that contains related trees but without the same predicate argument structure (files that contain all the adverb trees, etc.). If the word does not exist in the tree manager a default set of trees is returned. The trees can also be accessed by tree names,

Interface	Description
`word_to_morph()`	Given one token, returns its morphological records
`morph_to_syntax()`	Given morphological records of user inputs, returns their syntactical items
`get_feature_struct()`	Given feature identifier, returns feature structure object
`get_tag_tree()`	Given tree file name or tree name, returns tree object
`tree_lexicalization()`	Given morphological records, syntactic items, tree set and feature set, returns lexicalized TAG trees with feature structure

Figure 14. Forward Searching Interface Description

and in this case exactly one tree is returned if it is found. The interface for acquiring trees is defined as `get_tag_tree()`.

To lookup a particular lexicalized tree we search using a word or group of words. We then access the morphological records, syntactic items, tree set, and two feature sets (one from the morphological records, and another from the syntactic database) using the dataset. Still we need a procedure to combine them, producing a lexicalized TAG tree. This process is exactly what was called tree lexicalization in Section 2. Lexicalization is done in two steps.

The first step is to attach word(s) to TAG trees at the anchor nodes. Trees selected by the syntactic database possess one or more anchor nodes (represented by a diamond in the graphical view), and the number of anchor nodes is exactly the same as the words provided by the user. Tokens are attached to anchor nodes from left to right, i.e. the order they appear in trees is the same the order they are in user input.

The second step is to unify any feature structures that are to be merged into the lexicalized tree. Since we are dealing with both morphological features and syntactic features, they should be handled separately. Morphological features only carry inflectional information; therefore, they are unified into the bottom features of the anchor nodes respectively. In contrast, syntactic features capture lexical idiosyncrasies, and so they could be attached to any tree nodes. After this step, the result is a list of lexicalized trees with feature structures decorating each node in each elementary tree.

The procedure for final lexicalization using morphological and syntactical information is defined as `tree_lexicalization()`. And after this function returns, the data is returned to the client.

Interface	Description
`construct_tree_to_word_db()`	Constructs tree to words relations using syntactic database
`tree_to_word()`	Given TAG tree, returns words or phrases that could be used to lexicalized the tree

Figure 15. Backward Searching Interface Description

5.3. Backward Searching

Backward searching is invoked with an un-lexicalized TAG tree, and it returns words or phrases that could be used to lexicalize this tree. The data structure for backward searching is another mapping relation, which is constructed using the dataset in Section 3.

In essence, backward searching requires a database indexed by tree names, and the query returns a list of words and phrases. The mapping from tree names to words or phrases is constructed using only the syntactic database, since the mapping from words to tree files is established in syntactic database. In addition, because only tree files are involved in the mapping relation, all trees under the same file are mapped to the same set of words and phrases.

In order to construct a backward searching database, we need to iterate through each item in the syntactic database, using the tree file or family file in that item as the index, and append words or phrases in the syntactic item to the record in the backward searching database. The interface is defined as `construct_tree_to_word_db()`.

When the backend receives a request from the user to find all the words or phrases that could lexicalize an elementary tree, it queries the database with the tree file name to which the TAG tree in the request belongs, and returns the result. This interface is defined as: `tree_to_word()`.

5.4. Exploring The Tree Repository

The backend also allows requests be made for a single tree or for listing out all trees in a tree file. These two are usually needed when users are exploring the tree repository. Essentially, listing out trees using a file name is simply enumerates the name of the trees in that file, while querying for a single tree using a tree name retrieves that tree using the given name and parses it into a tree instance. This feature requires a database keeping the relationship between a tree name and a tree file name.

Although these two interfaces are straightforward, the tree returned is un-lexicalized, which means there are no feature structures and anchor words attached to it. However, for an un-lexicalized tree, backward searching should also be available, because

Interface	Description
`list_trees()`	Given a tree file name, returns all trees contained in the file
`get_tree()`	Given a tree name, return a TAG tree object

Figure 16. Exploring The Tree Repository Interface Description

all backward searching needs is the tree name, and we always know it.

5.5. Exception Handling

An exception is raised when the dataset appears to be inconsistent, or the backend could not find the desired entry. It interrupts the normal processing steps and then it tries to fix the problem, and then resume. By using *exception*, we do not always means the exception handling provided by programming language, but we refer to a more general situation, which indicates that something special must be done in order to complete the task.

There are several types of exceptions, and generally speaking, they could happen during any stage of processing. The most typical ones are, morphological record does not exist, syntactic item does not exist, and syntactic item exists but does not match. We will discuss them respectively.

By saying morphological record does not exist, we mean that a token from user input does not have a query result into morphological database. This can be quite common, especially when dealing with proper nouns. If it is this case, then a special default morphological record is used, which represents all tokens not in the morphological database for each part of speech tag.

In the second situation, "syntactic item does not exist", when a morphological record is passed to the interface for syntactic searching, if query returns no result, then the next step could not be taken, because of lack of information. In this case, the grammar viewer should return a special default syntactic item instead, which is designed for those unseen words in syntactic database, and has a default set of trees and features. It does not always yield good result, but is the best we could do.

In the last case, "syntactic item exists but does not match", we are in a situation very similar to the previous one. The major difference is that query to syntactic database does return something, but after comparing morphological information against their syntactic item identifiers (i.e. ENTRY and POS), there is not even one item that matches, and therefore the result is empty. To deal with this, the most straightforward method is to disable POS comparisons, matching only on the words.

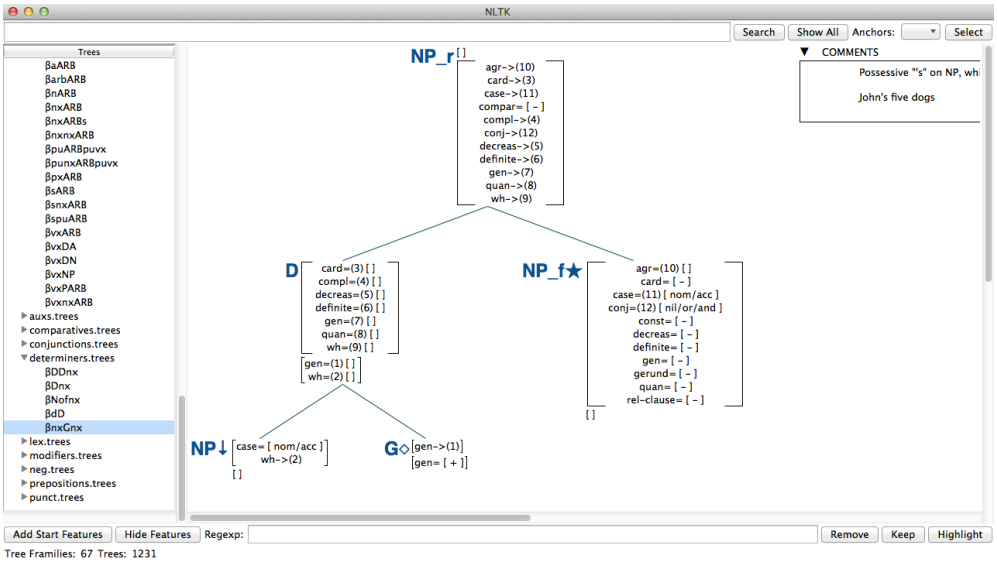


Figure 17. GUI to view the elementary trees in the XTAG grammar.

6. Graphical User Interface Design

Graphical User Interface (GUI), or called the front end, or the client, is the part that interacts with users, collecting user input and feedback, and encapsulates them into requests that will be sent to the server. The backend and GUI have complementary functionalities. In Section 5, we have seen that large portion of the backend contributes to its data processing ability. In this section, we will present knowledge about the GUI part of the grammar viewer.

The GUI is implemented using the Python GUI package Tkinter and uses additional helper functions from NLTK. The most useful feature of a GUI windows interface is that users can see the elementary TAG trees drawn on the canvas, and interact with them.

6.1. Window Layout

The main window layout is illustrated by Figure 17. It is divided into five sections, each being specialized for one or several similar purposes. The five sections are responsible for searching (the top), tree listing (middle-left), tree and feature drawing (middle-right), feature matching, displaying control (bottom), and status report (status bar) respectively.

On the top of the window, there are three buttons, one text box, and one drop-

down list. They help users to do both forward searching and backward searching. More specifically, the left half is used for forward searching, while the right half is used for backward searching. In order to do forward searching, users need to enter the word or phrase into the textbox, and then click "Search" button. After the backend has finished processing data, the result will be shown on the list box. After searching, if the user wants to restore the list box to the starting state, just click "Show All" button, and the list box will be redrawn to show all available trees and tree families. For backward searching, users could select from the dropdown list on the right part when there is a tree drawn on the canvas, and the list will be filled with all possible words and phrases that could be used to lexicalize the current displaying tree. Choose one word in the dropdown list and click "Select", then the word will be selected, and the effect is same as entering the selected word into the textbox and then click "Search".

On the middle left of the window, there is a list box. After application startup, the list box simply lists out all trees and tree families in a hierarchical manner, with tree file name being the first level label, and individual trees being the second level. If users would like to explore tree repository without lexicalizing any of them, they could began their journey from here. If the content of the list box has been changed due to a previous searching, users could restore it to the initial state by click the "Show All" button on the top of the window.

The middle right part of the window is the canvas dedicated to drawing trees and feature structures. If there are comments recorded in tree database, then they will also be drawn. The content on the canvas will change in response if user selects tree from the list box.

The bottom of the window is responsible for controlling feature structure display. There are five buttons and one textbox. The "Add Start Features" button on the left is used to attach the start feature in configuration file to the root node as its top feature. And the "Hide Features/Show Features" button is used to control whether to draw feature structure on the canvas. The three other buttons along with the textbox is used to match features on the current tree. The matching algorithm is presented in Section 5. The matching pattern is a regular expression that conforms to Python's re library, and we have three options: keep those matched, remove those matched, or simply highlight those matched. These three options correspond to the three buttons respectively.

A status report is shown in the status bar. In the current version we only display number of trees and tree families. However, future developers could in addition choose report the status of the morphological database, syntactic database and the feature template database.

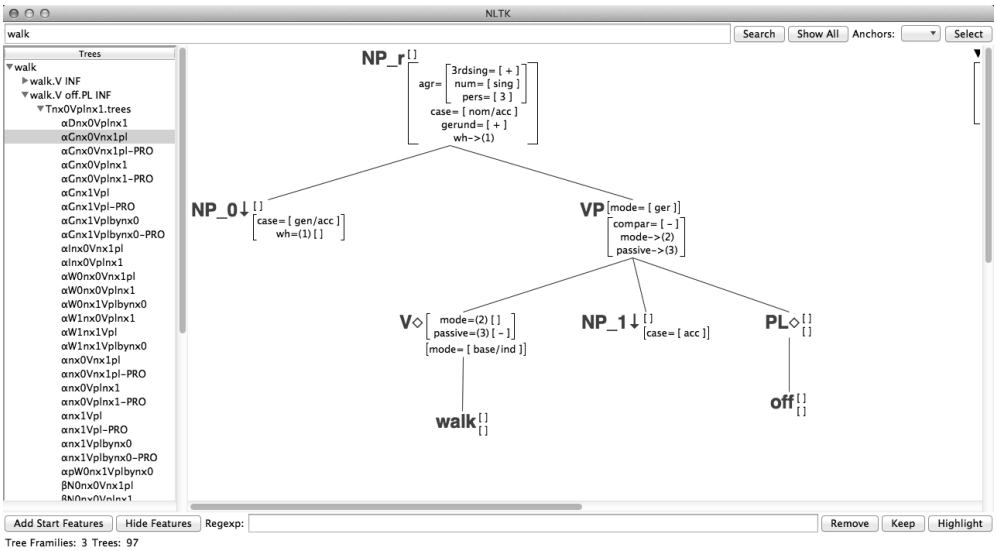


Figure 18. GUI to view lexicalized elementary trees in the XTAG grammar.

7. Conclusion

In this paper, we have described a Python-based interface for wide coverage Lexicalized Tree-adjoining Grammars. The grammars are part of the XTAG Grammar project and we hope that by providing an easy to use API to use these grammars that these grammars can be used for various NLP projects. We plan to incorporate our grammar viewer and the associated XTAG grammars for English and Korean into the Python Natural Language Toolkit (NLTK, `http://www.nltk.org/`).

Bibliography

Chen, John and K. Vijay-Shanker. Automated Extraction of TAGs from the Penn Treebank. In *Proc. of the 6th International Workshop on Parsing Technologies (IWPT-2000), Italy*, 2000.

Chiang, David. Statistical Parsing with an Automatically-Extracted Tree Adjoining Grammar. In *Proc. of ACL-2000*, 2000.

Joshi, Aravind and Yves Schabes. Tree adjoining grammars. In Rozenberg, G. and A. Salomaa, editors, *Handbook of Formal Languages and Automata*. Springer-Verlag, 1997.

Joshi, Aravind K, Leon S Levy, and Masako Takahashi. Tree Adjunct Grammars. *Journal of Computer and System Sciences*, 10(1):136–163, 1975. URL `http://dx.doi.org/10.1016/S0022-0000(75)80019-5`.

Rambow, O. and A. Joshi. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Wanner, Leo, editor, *Current Issues in Meaning-Text Theory*. Pinter, London, 1995.

Resnik, Philip. Probabilistic tree-adjoining grammars as a framework for statistical natural language processing. In *Proc. of COLING '92*, volume 2, pages 418–424, Nantes, France, 1992. URL `http://anthology.aclweb.org/C92-2065.pdf`.

Sarkar, Anoop. Applying co-training methods to statistical parsing. In *Proceedings of NAACL 2001*, Pittsburgh, PA, June 2001.

Schabes, Y. Stochastic lexicalized tree-adjoining grammars. In *Proc. of COLING '92*, volume 2, pages 426–432, Nantes, France, 1992. URL `http://dx.doi.org/10.3115/992133.992136`.

Schabes, Yves and Richard Waters. Tree insertion grammar: A cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4):479–513, 1995.

Schabes, Yves, Anne Abeillé, and Aravind K. Joshi. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August 1988.

Vijay-Shanker, K. *A Study of Tree Adjoining Grammars*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1987.

Xia, Fei. *Investigating the Relationship between Grammars and Treebanks for Natural languages*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 2001.

Xia, Fei, Chunghye Han, Martha Palmer, and Aravind Joshi. Automatically Extracting and Comparing Lexicalized Grammars for Different Languages. In *Proc. of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, Seattle, Washington, 2001.

XTAG Group. A lexicalized tree adjoining grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania, 2001.

Address for correspondence:
Anoop Sarkar
`anoop@sfu.ca`
School of Computing Science
Simon Fraser University
8888 University Drive, Burnaby, BC, Canada

PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 103 APRIL 2015

INSTRUCTIONS FOR AUTHORS

Manuscripts are welcome provided that they have not yet been published elsewhere and that they bring some interesting and new insights contributing to the broad field of computational linguistics in any of its aspects, or of linguistic theory. The submitted articles may be:

- long articles with completed, wide-impact research results both theoretical and practical, and/or new formalisms for linguistic analysis and their implementation and application on linguistic data sets, or
- short or long articles that are abstracts or extracts of Master's and PhD thesis, with the most interesting and/or promising results described. Also
- short or long articles looking forward that base their views on proper and deep analysis of the current situation in various subjects within the field are invited, as well as
- short articles about current advanced research of both theoretical and applied nature, with very specific (and perhaps narrow, but well-defined) target goal in all areas of language and speech processing, to give the opportunity to junior researchers to publish as soon as possible;
- short articles that contain contraversing, polemic or otherwise unusual views, supported by some experimental evidence but not necessarily evaluated in the usual sense are also welcome.

The recommended length of long article is 12–30 pages and of short paper is 6–15 pages.

The copyright of papers accepted for publication remains with the author. The editors reserve the right to make editorial revisions but these revisions and changes have to be approved by the author(s). Book reviews and short book notices are also appreciated.

The manuscripts are reviewed by 2 independent reviewers, at least one of them being a member of the international Editorial Board.

Authors receive a printed copy of the relevant issue of the PBML together with the original pdf files.

The guidelines for the technical shape of the contributions are found on the web site `http://ufal.mff.cuni.cz/pbml`. If there are any technical problems, please contact the editorial staff at `pbml@ufal.mff.cuni.cz`.