Inducing a multilingual dictionary from a parallel multitext in related languages

Dmitriy Genzel
Department of Computer Science
Box 1910
Brown University
Providence, RI 02912, USA
dg@cs.brown.edu

Abstract

Dictionaries and word translation models are used by a variety of systems, especially in machine translation. We build a multilingual dictionary induction system for a family of related resource-poor languages. We assume only the presence of a single medium-length multitext (the Bible). The techniques rely upon lexical and syntactic similarity of languages as well as on the fact that building dictionaries for several pairs of languages provides information about other pairs.

1 Introduction and Motivation

Modern statistical natural language processing techniques require large amounts of human-annotated data to work well. For practical reasons, the required amount of data exists only for a few languages of major interest, either commercial or governmental. As a result, many languages have very little computational research done in them, especially outside the borders of the countries in which these languages are spoken. Some of these languages are, however, major languages with hundreds of millions of speakers. Of the top 10 most spoken languages, Linguistic Data Consortium at University of Pennsylvania, the premier U.S. provider of corpora, offers text corpora only in 7 (The World Factbook (2004), 2000 estimate) Only a few of the other languages (French, Arabic, and Czech) have resources provided by LDC. Many Asian and Eastern European languages number tens of millions of speakers, yet very few of these seem to have any related computational linguistics work, at least as presented at the international conferences, such as the ACL.[1]

The situation is not surprising, nor is it likely to significantly change in the future. Luckily, most of these less-represented languages belong to language families with several prominent members. As a result, some of these languages have siblings with more resources and published research. [2] Interestingly, the better-endowed siblings are not always the ones with more native speakers, since political considerations are often more important.[3] If one is able to use the resources available in one language (henceforth referred to as *source*) to facilitate the creation of tools and resource in another, related language (*target*), this problem would be alleviated. This is the ultimate goal of this project, but in the first stage we focus on multi-language dictionary induction.

Building a high-quality dictionary, or even better, a joint word distribution model over all the languages in a given family is very important, because using such a model one can use a variety of techniques to project information across languages, e.g. to parse or to translate. Building a unified model for more than a pair of languages improves the quality over building several unrelated pairwise models, because relating them to each other provides additional information. If we know that word a in language A has as its likely translation word b in language B, and b is translated as c in C, then we also know that a is likely to be translated as c, without looking at

[1]The search through ACL Anthology, for e.g., Telugu ($\sim$70 million speakers) shows only casual mention of the language.

[2]Telugu's fellow Dravidian language *Tamil* ($\sim$65 million speakers) has seen some papers at the ACL

[3]This is the case with Tamil vs. Telugu.

the A to C model.

2 Previous Work

There has been a lot of work done on building dictionaries, by using a variety of techniques. One good overview is Melamed (2000). There is work on lexicon induction using string distance or other phonetic/orthographic comparison techniques, such as Mann and Yarowsky (2001) or semantic comparison using resources such as WordNet (Kondrak, 2001). Such work, however, primarily focuses on finding cognates, whereas we are interested in translations of all words. Moreover, while some techniques (e.g., Mann and Yarowsky (2001)) use multiple languages, the languages used *have* resources such as dictionaries between some language pairs. We do not require any dictionaries for any language pair.

An important element of our work is focusing on more than a pair of languages. There is an active research area focusing on multi-source translation (e.g., Och and Ney (2001)). Our setting is the reverse: we do not use multiple dictionaries in order to translate, but translate (in a very crude way) in order to build multiple dictionaries.

Many machine translation techniques require dictionary building as a step of the process, and therefore have also attacked this problem. They use a variety of approaches (a good overview is Koehn and Knight (2001)), many of which require advanced tools for both languages which we are not able to use. They also use bilingual (and to some extent monolingual) corpora, which we do have available. They do not, however, focus on related languages, and tend to ignore lexical similarity [4], nor are they able to work on more than a pair of languages at a time.

It is also worth noting that there has been some MT work on related languages which explores language similarity in an opposite way: by using dictionaries and tools for both languages, and assuming that a near word-for-word approach is reasonable (Hajic et al., 2000).

[4]Much of recent MT research focuses on pairs of languages which are not related, such as English-Chinese, English-Arabic, etc.

3 Description of the Problem

Let us assume that we have a group of related languages, $L_1 \ldots L_n$, and a parallel sentence-aligned multitext C, with corresponding portions in each language denoted as $C_1 \ldots C_n$. Such a multitext exists for virtually all the languages in the form of the Bible. Our goal is to create a multilingual dictionary by learning the joint distribution $P(x_1 \ldots x_n)_{x_i \in L_i}$ which is simply the expected frequency of the n-tuple of words in a completely word-aligned multitext. We will approach the problem by learning pairwise language models, although leaving some parameters free, and then combine the models and learn the remaining free parameters to produce the joint model.

Let us, therefore, assume that we have a set of models $\{P(x, y|\theta_{ij})_{x \in L_i, y \in L_j}\}_{i \neq j}$ where θij is a parameter vector for pairwise model for languages L_i and L_j. We would like to learn how to combine these models in an optimal way. To solve this problem, let us first consider a simpler and more general setting.

3.1 Combining Models of Hidden Data

Let X be a random variable with distribution $P_{\text{true}}(x)$, such that no direct observations of it exist. However, we may have some indirect observations of X and have built several models of X's distribution, $\{P_i(x|\theta_i)\}_{i=1}^n$, each parameterized by some parameter vector θ_i. P_i also depends on some other parameters that are fixed. It is important to note that the space of models obtained by varying θ_i is only a small subspace of the probability space. Our goal is to find a good estimate of $P_{\text{true}}(x)$.

The main idea is that if some P_i and P_j are close (by some measure) to P_{true}, they have to be close to each other as well. We will therefore make the assumption that if some models of X are close to each other (and we have reason to believe they are fair approximations of the true distribution) they are also close to the true distribution. Moreover, we would like to set the parameters θ_i in such a way that $P(x_i|\theta_i)$ is as close to the other models as possible. This leads us to look for an estimate that is as close to all of our models as possible, under the optimal values of θ_i's, or more formally:

$$P_{\text{est}} = \arg\min_{\hat{P}(\cdot)} \min_{\theta_1} \dots \min_{\theta_n} d(\hat{P}(\cdot), P_1(\cdot|\theta_1), \dots P_n(\cdot|\theta_n))$$

where d measures the distance between $\hat{P}$ and all the P_i under the parameter setting θ_i. Since we have no reason to prefer any of the P_i, we choose the following symmetric form for d:

$$\sum_{i=1}^{n} D(\hat{P}(\cdot)||P_i(\cdot|\theta_i))$$

where D is a reasonable measure of distance between probability distributions. The most appropriate and the most commonly used measure in such cases in the Kullback-Leibler divergence, also known as relative entropy:

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

It turns out that it is possible to find the optimal $\hat{P}$ under these circumstances. Taking a partial derivative and solving, we obtain:

$$\hat{P}(x) = \frac{\prod_{i=1}^{n} P_i(x|\theta_i)^{1/n}}{\sum_{x' \in X} \prod_{i=1}^{n} P_i(x'|\theta_i)^{1/n}}$$

Substituting this value into the expression for function d, we obtain the following distance measure between the P_i's:

$$\begin{aligned} & d'(P_1(X|\theta_1) \dots P_n(X|\theta_n)) \\ & \quad = \min_{\hat{P}} d(\hat{P}, P_1(X|\theta_1), \dots P_n(X|\theta_n)) \\ & \quad = -\log \sum_{x \in X} \prod_{i=1}^{n} P_i(x|\theta_i)^{1/n} \end{aligned}$$

This function is a generalization of the well-known Bhattacharyya distance for two distributions (Bhattacharyya, 1943):

$$b(p,q) = \sum_i \sqrt{p_i q_i}$$

These results suggest the following **Algorithm 1** to optimize d (and d'):

```
• Set all $\theta_i$ randomly
• Repeat until change in $d$ is very small:
    – Compute $\hat{P}$ according to the above formula
    – For $i$ from 1 to $n$
        * Set $\theta_i$ in such a way as to minimize $D(\hat{P}(X)||P_i(X|\theta_i))$
    – Compute $d$ according to the above formula
```

Each step of the algorithm minimizes d. It is also easy to see that minimizing $D(\hat{P}(X)||P_i(X|\theta_i))$ is the same as setting the parameters θ_i in order to maximize $\prod_{x \in X} P_i(x|\theta_i)^{\hat{P}(x)}$, which can be interpreted as maximizing the probability under P_i of a corpus in which word x appears $\hat{P}(x)$ times. In other words, we are now optimizing $P_i(X)$ given an observed corpus of X, which is a much easier problem. In many types of models for P_i the Expectation-Maximization algorithm is able to solve this problem.

3.2 Combining Pairwise Models

Following the methods outlined in the previous section, we can find an optimal joint probability $P(x_1 \dots x_n)_{x_i \in L_i}$ if we are given several models $P_j(x_1 \dots x_n|\theta_j)$. Instead, we have a number of pairwise models. Depending on which independence assumptions we make, we can define a joint distribution over all the languages in various ways. For example, for three languages, A, B, and C, and we can use the following set of models:

$$\begin{aligned} P_1(A,B,C) &= P(A|B)P(B|C)P(C) \\ P_2(A,B,C) &= P(C|A)P(A|B)P(B) \\ P_3(A,B,C) &= P(B|C)P(C|A)P(A) \end{aligned}$$

and

$$\begin{aligned} & d'(\hat{P}, P_1, P_2, P_3) \\ & \quad = D(\hat{P}||P_1) + D(\hat{P}||P_2) + D(\hat{P}||P_3) \\ & \quad = 2H(\hat{P}(A,C), P(A,C)) \\ & \quad + 2H(\hat{P}(A,B), P(A,B)) \\ & \quad + 2H(\hat{P}(B,C), P(B,C)) - 3H(\hat{P}) \\ & \quad - H(\hat{P}(A), P(A)) - H(\hat{P}(B), P(B)) \\ & \quad - H(\hat{P}(C), P(C)) \end{aligned}$$

where $H(\cdot)$ is entropy, $H(\cdot,\cdot)$ is cross-entropy, and $\hat{P}(A,B)$ means $\hat{P}$ marginalized to variables A, B. The last three cross-entropy terms involve monolingual models which are not parameterized. The entropy term does not involve any of the pairwise distributions. Therefore, if $\hat{P}$ is fixed, to maximize d'

we need to maximize each of the bilingual cross-entropy terms.

This means we can apply the algorithm from the previous section with a small modification (**Algorithm 2**):

- Set all θ_{ij} (for each language pair i, j) randomly

- Repeat until change in d is very small:
  - Compute P_i for $i = 1 \dots k$ where k is the number of the joint models we have chosen
  - Compute $\hat{P}$ from $\{P_i\}$
  - For i, j such that $i \neq j$
    - Marginalize $\hat{P}$ to (L_i, L_j)
    - Set θ_{ij} in such a way as to minimize $D(\hat{P}(L_i, L_j) || P_i(L_i, L_j | \theta_{ij}))$
  - Compute d according to the above formula

Most of the θ parameters in our models can be set by performing EM, and the rest are discrete with only a few choices and can be maximized over by trying all combinations of them.

4 Building Pairwise Models

We now know how to combine pairwise translation models with some free parameters. Let us now discuss how such models might be built.

Our goal at this stage is to take a parallel bitext in related languages A and B and produce a joint probability model $P(x, y)$, where $x \in A, y \in B$. Equivalently, since the models $P_A(x)$ and $P_B(y)$ are easily estimated by maximum likelihood techniques from the bitext, we can estimate $P_{A\to B}(y|x)$ or $P_{B\to A}(x|y)$. Without loss of generality, we will build $P_{A\to B}(y|x)$.

The model we are building will have a number of free parameters. These parameters will be set by the algorithm discussed above. In this section we will assume that the parameters are fixed.

Our model is a mixture of several components, each discussed in a separate section below:

$$\begin{aligned} P_{A\to B}(y|x) &= \lambda_{fw}(x) P_{fwA\to B}(y|x) \\ &+ \lambda_{bw}(x) P_{bwA\to B}(y|x) \\ &+ \lambda_{char}(x) P_{charA\to B}(y|x) \\ &+ \lambda_{pref}(x) P_{prefA\to B}(y|x) \\ &+ \lambda_{suf}(x) P_{sufA\to B}(y|x) \\ &+ \lambda_{cons}(x) P_{consA\to B}(y|x) \end{aligned} \tag{1}$$

where all λs sum up to one. The λs are free parameters, although to avoid over-training we tie the λs for x's with similar frequencies. These lambdas form a part of the θ_{ij} parameter mentioned previously, where $L_i = A$ and $L_j = B$.

The components represent various constraints that are likely to hold between related languages.

4.1 GIZA (forward)

This component is in fact GIZA++ software, originally created by John Hopkins University's Summer Workshop in 1999, improved by Och (2000). This software can be used to create word alignments for sentence-aligned parallel corpora as well as to induce a probabilistic dictionary for this language pair.

The general approach taken by GIZA is as follows. Let L_A and L_B be the portions of the parallel text in languages A and B respectively, and $L_A = (x_i)_{i=1\dots n}$ and $L_B = (y_i)_{i=1\dots m}$. We can define $P(L_B|L_A)$ as

$$\max_{P_{A\to B}} \max_{P_{\text{aligns}}} \sum_{i=1}^{n} \sum_{j=1}^{m} P_{A\to B}(y_j|x_i)\, P_{\text{aligns}}(x_i|j)$$

The GIZA software does the maximization by building a variety of models, mostly described by Brown et al. (1993). GIZA can be tuned in various ways, most importantly by choosing which models to run and for how many iterations. We treat these parameters as free, to be set along with the rest at a later stage.

As a side effect of GIZA's optimization, we obtain the $P_{A\to B}(y|x)$ that maximizes the above expression. It is quite reasonable to believe that a model of this sort is also a good model for our purposes. This model is what we refer to as $P_{fwA\to B}(y|x)$ in the model overview.

GIZA's approach is not, however, perfect. GIZA builds several models, some quite complex, yet it does not use all the information available to it, notably the lexical similarity between the languages. Furthermore, GIZA tries to map words (especially rare ones) into other words if possible, even if the sentence has no direct translation for the word in question.

These problems are addressed by using other models, described in the following sections.

4.2 GIZA (backward)

In the previous section we discussed using GIZA to try to optimize $P(L_B|L_A)$. It is, however, equally reasonable to try to optimize $P(L_A|L_B)$ instead. If we do so, we can obtain $P_{fwB\to A}(x|y)$ that produces maximal probability for $P(L_A|L_B)$. We, however need a model of $P_{A\to B}(y|x)$. This is easily obtained by using Bayes' rule:

$$P_{bwA\to B}(y|x) = \frac{P_{fwB\to A}(x|y)P_B(y)}{P_A(x)}$$

which requires us to have $P_B(y)$ and $P_A(x)$. These models can be estimated directly from L_B and L_A, by using maximum likelihood estimators:

$$P_A(x) = \frac{\sum_i \delta(x_i, x)}{n}$$

and

$$P_B(y) = \frac{\sum_i \delta(y_i, y)}{m}$$

where $\delta(x, y)$ is the Kronecker's delta function, which is equal to 1 if its arguments are equal, and to 0 otherwise.

4.3 Character-based model

This and the following models all rely on having a model of $P_{A\to B}(y|x)$ to start from. In practice it means that this component is estimated following the previous components and uses the models they provide as a starting point.

The basic idea behind this model is that in related languages words are also related. If we have a model P_c of translating characters in language A into characters in language B, we can define the model for translating entire words.

Let word x in language A consists of characters x_1 through x_n, and word y in language B consist of characters y_1 through y_m.

Let us define (the unnormalized) character model:

$$P_{uchar}(y|x) = P_{charlen}(y|x, m)P_{length}(m|x)$$

i.e., estimating the length of y first, and y itself afterward. We make an independence assumption that the length of y depends only on length of x, and are able to estimate the second term above easily. The first term is harder to estimate.

First, let us consider the case where lengths of x and y are the same ($m = n$). Then,

$$P_{charlen}(y|x, n) = \prod_{i=1}^{n} P_c(y_i|x_i)$$

Let y^j be word y with j's character removed. Let us now consider the case when $m > n$. We define (recursively):

$$P_{charlen}(y|x, m) = \sum_{i=1}^{m} \frac{1}{m} P_{charlen}(y^i|x, m-1)$$

Similarly, if $n > m$:

$$P_{charlen}(y|x) = \sum_{i=1}^{n} \frac{1}{n} P_{charlen}(y|x^i, m)$$

It is easy to see that this is a valid probability model over all sequences of characters. However, y is not a random sequence of characters, but a word in language B, moreover, it is a word that can serve as a potential translation of word x. So, to define a proper distribution over words y given a word x and a set of possible translations of x, $T(x)$

$$\begin{aligned} P_{char}(y|x) &= P_{uchar}\,(y|x, y \in T(x)) \\ &= \delta_{y' \in T(x)} \frac{P_{uchar}(y, y \in T(x)|x)}{\sum_{y' \in T(x)} P_{uchar}(y'|x)} \end{aligned}$$

This is the complete definition of P_{char}, except for the fact that we are implicitly relying upon the character-mapping model, P_c, which we need to somehow obtain. To obtain it, we rely upon GIZA again. As we have seen, GIZA can find a good word-mapping model if it has a bitext to work from. If we have a $P_{A\to B}$ word-mapping model of some sort, it is equivalent to having a parallel bitext with words y and x treated as a sequence of characters, instead of indivisible tokens. Each (x, y) word pair would occur $P_{A\to B}(x, y)$ times in this corpus. GIZA would then provide us with the P_c model we need, by optimizing the probability B language part of the model given the language A part.

4.4 Prefix Model

This model and the two models that follow are built on the same principle. Let there be a function $f : A \to C_A$ and a function $g : B \to C_B$. These functions group words in A and B into some finite set of classes. If we have some $P_{A\to B}(y|x)$ to start with, we can define

$$\begin{aligned} & P_{fgA\to B}(y|x) \\ = \quad & P\left(y|g(y)\right) P\left(g(y)|f(x)\right) P\left(f(x)|x\right) \\ = \quad & P(y) \frac{\sum_{x':f(x')=f(x)} \sum_{y':g(y')=g(y)} P(x',y')}{\left(\sum_{x':f(x')=f(x)} P(x')\right)\left(\sum_{y':g(y')=g(y)} P(y')\right)} \end{aligned}$$

For the prefix model, we rely upon the following idea: words that have a common prefix often tend to be related. Related words probably should translate as related words in the other language as well. In other words, we are trying to capture word-level semantic information. So we define the following set of f and g functions:

$$f_n(x) = \text{prefix}(x, n)$$

$$g_m(y) = \text{prefix}(y, m)$$

where n and m are free parameters, whose values we will determine later. We therefore define $P_{prefA\to B}$ as P_{fg} with f and g specified above.

4.5 Suffix Model

Similarly to a prefix model mentioned above, it is also useful to have a suffix model. Words that have the same suffixes are likely to be in the same grammatical case or share some morphological feature which may persist across languages. In either case, if a strong relationship exists between the resulting classes, it provides good evidence to give higher likelihood to the word belonging to these classes. It is worth noting that this feature (unlike the previous one) is unlikely to be helpful in a setting where languages are not related.

The functions f and g are defined based on a set of suffixes S_A and S_B which are learned automatically. $f(x)$ is defined as the longest possible suffix of x that is in the set S_A, and g is defined similarly, for S_B.

The sets S_A and S_B are built as follows. We start with all one-character suffixes. We then consider two-letter suffixes. We add a suffix to the list if it occurs much more often than can be expected based on the frequency of its first letter in the penultimate position, times the frequency of its second letter in the last position. We then proceed in a similar way for three-letter suffixes. The threshold value is a free parameter of this model.

4.6 Constituency Model

If we had information about constituent boundaries in either language, it would have been useful to make a model favoring alignments that do not cross constituent boundaries. We do not have this information at this point. We can assume, however, that any sequence of three words is a constituent of sorts, and build a model based on that assumption.

As before, let $L_A = (x_i)_{i=1\ldots n}$ and $L_B = (y_i)_{i=1\ldots m}$. Let us define as $C_A(i)$ a triple of words (x_{i-1}, x_i, x_{i+1}) and as $C_B(j)$ a triple (y_{j-1}, y_j, y_{j+1}). If we have some model $P_{A\to B}$, we can define

$$\begin{aligned} P_{C_A\to C_B}(j|i) \quad = \quad & \tfrac{1}{C} P_{A\to B}(y_{j-1}|x_{i-1}) P_{A\to B}(y_j|x_i) \\ \times \quad & P_{A\to B}(y_{j+1}|x_{i+1}) \end{aligned}$$

where C is the sum over j of the above products, and serves to normalize the distribution.

$$\begin{aligned} & P_{consA\to B}(y|x) \\ = \quad & \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} P(y|C_B(j)) P_{C_A\to C_B}(j|i) P(C_A(i)|x) \\ = \quad & \textstyle\sum_{i:x_i=x} \sum_{j=1}^{m} P(y|C_B(j)) P_{C_A\to C_B}(j|i) \\ = \quad & \frac{1}{\sum_{j=1} \delta(y_j,y)} \textstyle\sum_{i:x_i=x} \sum_{j:y_i=y} P_{C_A\to C_B}(j|i) \end{aligned}$$

5 Evaluation

The output of the system so far is a multi-lingual word translation model. We will evaluate it by producing a tri-lingual dictionary (Russian-Ukrainian-Belorussian), picking a highest probability translation for each word, from the corresponding Bibles. Unfortunately, we do not have a good hand-built trilingual dictionary to compare it to, but only one good bilingual one, Russian-Ukrainian[5]. We will therefore take the Russian-Ukrainian portion of our dictionary and compare it to the hand-built one.

Our evaluation metric is the number of entries that match between these dictionaries. If a word has several translations in the hand-built dictionary, match-

[5]The lack of such dictionaries is precisely *why* we do this work

ing any of them counts as correct. It is worth noting that for all the dictionaries we generate, the total number of entries is the same, since all the words that occur in the source portion of the corpus have an entry. In other words, precision and recall are proportional to each other and to our evaluation metric.

Not all of the words that occur in our dictionary occur in the hand-built dictionary and vice versa. An absolute upper limit of performance, therefore, for this evaluation measure is the number of left-hand-side entries that occur in both dictionaries.

In fact, we cannot hope to achieve this number. First, because the dictionary translation of the word in question might never occur in the corpus. Second, even if it does, but never co-occurs in the same sentence as its translation, we will not have any basis to propose it as a translation.[6]. Therefore we have a "achievable upper limit", the number of words that have their "correct" translation co-occur at least once. We will compare our performance to this upper limit.

Since there is no manual tuning involved we do not have a development set, and use the whole bible for training (the dictionary is used as a test set, as described above).

We evaluate the performance of the model with just the GIZA component as the baseline, and add all the other components in turn. There are two possible models to evaluate at each step. The pairwise model is the model given in equation 1 under the parameter setting given by Algorithm 2, with Belorussian used as a third language. The joint model is the full model over these three languages as estimated by Algorithm 2. In either case we pick a highest probability Ukrainian word as a translation of a given Russian word.

The results for Russian-Ukrainian bibles are presented in Table 1. The "oracle" setting is the setting obtained by tuning on the test set (the dictionary). We see that using a third language to tune works just as well, obtaining the true global maximum for the model. Moreover, the joint model (which is more flexible than the model in Equation 1) does even better. This was unexpected for us, because the joint model relies on three pairwise models equally, and Russian-Belorussian and Ukrainian-Belorussian models are bound to be less reliable for Russian-Ukrainian evaluation. It appears, however, that our Belorussian bible is translated directly from Russian rather than original languages, and parallels Russian text more than could be expected.

[6]Strictly speaking, we might be able to infer the word's existence in some cases, by performing morphological analysis and proposing a word we have not seen, but this seems too hard at the moment

Table 1: Evaluation for Russian-Ukrainian (with Belorussian to tune)

Stage	**Pair**	**Joint**
Forward (baseline)	62.3%	71.7%
Forward+chars	77.1%	84.2%
Forward+chars+backward	81.3%	84.1%
Fw+chars+bw+prefix	83.5%	84.5%
Fw+chars+bw+prefix+suffix	84.5%	85%
Fw+chars+bw+pref+suf+const	84.5%	85.2%
"Oracle" setting for λ's	84.6%	

Table 2: Evaluation for Russian-Ukrainian (with Belorussian and Polish)

Tuned by	**Pair**	**Joint**
Belorussian (prev. table)	84.5%	85.2% &
Polish	84.6%	78.6%
Both	84.5%	85.2%
"Oracle" tuning	84.5%	

To insure our results are not affected by this fact we also try Polish separately and in combination with Belorussian (i.e. a model over 4 languages), as shown in Table 2.

These results demonstrate that the joint model is not as good for Polish, but it still finds the optimal parameter setting. This leads us to propose the following extension: let us marginalize joint Russian-Ukrainian-Belorussian model into just Russian-Ukrainian, and add this model as yet another component to Equation 1. Now we cannot use Belorussian as a third language, but we can use Polish, which we know works just as well for tuning. The resulting performance for the model is **85.7%**, our best result to date.

6 Discussion and Future Work

We have built a system for multi-dictionary induction from parallel corpora which significantly improves quality over the standard existing tool (GIZA) by taking advantage of the fact that languages are related and we have a group of more than two of them. Because the system attempts to be completely agnostic about the languages it works on, it might be used successfully on many language groups, requiring almost no linguistic knowledge on the part of the user. Only the prefix and suffix components are somewhat language-specific, but even they are sufficiently general to work, with varying degree of success, on most inflective and agglutinative languages (which form a large majority of languages). For generality, we would also need a model of infixes, for languages such as Hebrew or Arabic. We must admit, however, that we have not tested our approach on other language families yet. It is our short term plan to test our model on several Romance languages, e.g. Spanish, Portuguese, French.

Looking at the first lines of Table 1, one can see that using more than a pair of languages with a model using only a small feature set can dramatically improve performance (compare second and third columns), while able to find the optimal values for all internal parameters.

As discussed in the introduction, the ultimate goal of this project is to produce tools, such as a parser, for languages which lack them. Several approaches are possible, all involving the use of the dictionary we built. While working on this project, we would no longer be treating all languages in the same way. We would use the tools available for that language to further improve the performance of pairwise models involving that language and, indirectly, even the pairs not involving this language. Using these tools, we may be able to improve the word translation model even further, simply as a side effect.

Once we build a high-quality dictionary for a special domain such as the Bible, it might be possible to expand to a more general setting by mining the Web for potential parallel texts.

Our technique is limited in the coverage of the resulting dictionary which can only contain words which occur in our corpus. Whatever the corpus may be, however, it will include the most common words in the target language. These are the words that tend to vary the most between related (and even unrelated) languages. The relatively rare words (e.g. domain-specific and technical terms) can often be translated simply by inferring morphological rules transforming words of one language into another. Thus, one may expand the dictionary coverage using non-parallel texts in both languages, or even in just one language if its morphology is sufficiently regular.

References

The Central Intelligence Agency. 2004. The world factbook.

A. Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109.

P.F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

J. Hajic, J. Hric, and V. Kubon. 2000. Machine translation of very close languages. In *Proccedings of the 6th Applied Natural Language Processing Conference*.

P. Koehn and K. Knight. 2001. Knowledge sources for word-level translation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

G. Kondrak. 2001. Identifying cognates by phonetic and semantic similarity. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, PA*, pages 103–110.

G. Mann and D. Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh, PA*, pages 151–158.

I. D. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26:221–249, June.

F. J. Och and H. Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hongkong, China, October.

F. J. Och and H. Ney. 2001. Statistical multi-source translation. In *Proccedings of MT Summit VIII*, pages 253–258.