Simple Effective Decipherment via Combinatorial Optimization

Taylor Berg-Kirkpatrick and **Dan Klein**
Computer Science Division
University of California at Berkeley
{tberg, klein}@cs.berkeley.edu

Abstract

We present a simple objective function that when optimized yields accurate solutions to both decipherment and cognate pair identification problems. The objective simultaneously scores a matching between two alphabets and a matching between two lexicons, each in a different language. We introduce a simple coordinate descent procedure that efficiently finds effective solutions to the resulting combinatorial optimization problem. Our system requires only a list of words in both languages as input, yet it competes with and surpasses several state-of-the-art systems that are both substantially more complex and make use of more information.

1 Introduction

Decipherment induces a correspondence between the words in an unknown language and the words in a known language. We focus on the setting where a close correspondence between the alphabets of the two languages exists, but is unknown. Given only two lists of words, the lexicons of both languages, we attempt to induce the correspondence between alphabets and identify the cognates pairs present in the lexicons. The system we propose accomplishes this by defining a simple combinatorial optimization problem that is a function of both the alphabet and cognate matchings, and then induces correspondences by optimizing the objective using a block coordinate descent procedure.

There is a range of past work that has variously investigated cognate detection (Kondrak, 2001; Bouchard-Côté et al., 2007; Bouchard-Côté et al., 2009; Hall and Klein, 2010), character-level decipherment (Knight and Yamada, 1999; Knight et al., 2006; Snyder et al., 2010; Ravi and Knight, 2011), and bilingual lexicon induction (Koehn and Knight, 2002; Haghighi et al., 2008). We consider a common element, which is a model wherein there are character-level correspondences and word-level correspondences, with the word matching parameterized by the character one. This approach subsumes a range of past tasks, though of course past work has specialized in interesting ways.

Past work has emphasized the modeling aspect, where here we use a parametrically simplistic model, but instead emphasize inference.

2 Decipherment as Two-Level Optimization

Our method represents two matchings, one at the alphabet level and one at the lexicon level. A vector of variables x specifies a matching between alphabets. For each character i in the source alphabet and each character j in the target alphabet we define an indicator variable x_{ij} that is on if and only if character i is mapped to character j. Similarly, a vector y represents a matching between lexicons. For word u in the source lexicon and word v in the target lexicon, the indicator variable y_{uv} denotes that u maps to v. Note that the matchings need not be one-to-one.

We define an objective function on the matching variables as follows. Let $\textsc{EditDist}(u, v; x)$ denote the edit distance between source word u and target word v given alphabet matching x. Let the length of word u be l_u and the length of word w be l_w. This edit distance depends on x in the following way. Insertions and deletions always cost a constant ϵ.[1] Substitutions also cost ϵ unless the characters are matched in x, in which case the substitution is

[1] In practice we set $\epsilon = \frac{1}{l_u + l_v}$. $l_u + l_v$ is the maximum number of edit operations between words u and v. This normalization insures that edit distances are between 0 and 1 for all pairs of words.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 313–321,
Edinburgh, Scotland, UK, July 27–31, 2011.

free. Now, the objective that we will minimize can be stated simply: $\sum_u \sum_v y_{uv} \cdot \text{EDITDIST}(u, v; x)$, the sum of the edit distances between the matched words, where the edit distance function is parameterized by the alphabet matching.

Without restrictions on the matchings x and y this objective can always be driven to zero by either mapping all characters to all characters, or matching none of the words. It is thus necessary to restrict the matchings in some way. Let I be the size of the source alphabet and J be the size of the target alphabet. We allow the alphabet matching x to be many-to-many but require that each character participate in no more than two mappings and that the total number of mappings be $\max(I, J)$, a constraint we refer to as *restricted-many-to-many*. The requirements can be encoded with the following linear constraints on x:

$$\forall_i \sum_j x_{ij} \leq 2$$
$$\forall_j \sum_i x_{ij} \leq 2$$
$$\sum_i \sum_j x_{ij} = \max(I, J)$$

The lexicon matching y is required to be τ*-one-to-one*. By this we mean that y is an at-most-one-to-one matching that covers proportion τ of the smaller of the two lexicons. Let U be the size of the source lexicon and V be this size of the target lexicon. This requirement can be encoded with the following linear constraints:

$$\forall_u \sum_v y_{uv} \leq 1$$
$$\forall_v \sum_u y_{uv} \leq 1$$
$$\sum_u \sum_v y_{uv} = \tau \min(U, V)$$

Now we are ready to define the full optimization problem. The first formulation is called the *Implicit Matching Objective* since includes an implicit minimization over edit alignments inside the computation of EDITDIST.

(1) *Implicit Matching Objective*:

$$\min_{x,y} \sum_u \sum_v y_{uv} \cdot \text{EDITDIST}(u, v; x)$$
$$\text{s.t. } x \text{ is restricted-many-to-many}$$
$$y \text{ is } \tau\text{-one-to-one}$$

In order to get a better handle on the shape of the objective and to develop an efficient optimization procedure we decompose each edit distance computation and re-formulate the optimization problem in Section 2.2.

2.1 Example

Figure 1 presents both an example matching problem and a diagram of the variables and objective. Here, the source lexicon consists of the English words `(cat, bat, cart, rat, cab)`, and the source alphabet consists of the characters `(a, b, c, r, t)`. The target alphabet is `(0, 1, 2, 3)`. We have used digits as symbols in the target alphabet to make it clear that we treat the alphabets as disjoint. We have no prior knowledge about any correspondence between alphabets, or between lexicons.

The target lexicon consists of the words `(23, 1233, 120, 323, 023)`. The bipartite graphs show a specific setting of the matching variables. The bold edges correspond to the x_{ij} and y_{uv} that are one. The matchings shown achieve an edit distance of zero between all matched word pairs except for the pair `(cat, 23)`. The best edit alignment for this pair is also diagrammed. Here, '`a`' is aligned to '`2`', '`t`' is aligned to '`3`', and '`c`' is deleted and therefore aligned to the null position '`#`'. Only the initial deletion has a non-zero cost since all other alignments correspond to substitutions between characters that are matched in x.

2.2 Explicit Objective

Computing $\text{EDITDIST}(u, v; x)$ requires running a dynamic program because of the unknown edit alignments; here we define those alignments z explicitly, which makes the $\text{EDITDIST}(u, v; x)$ easy to write explicitly at the cost of more variables. However, by writing the objective in an explicit form that refers to these edit variables, we are able to describe a efficient block coordinate descent procedure that can be used for optimization.

$\text{EDITDIST}(u, v; x)$ is computed by minimizing over the set of monotonic alignments between the characters of the source word u and the characters of the target word v. Let u_n be the character at the nth position of the source word u, and similarly for

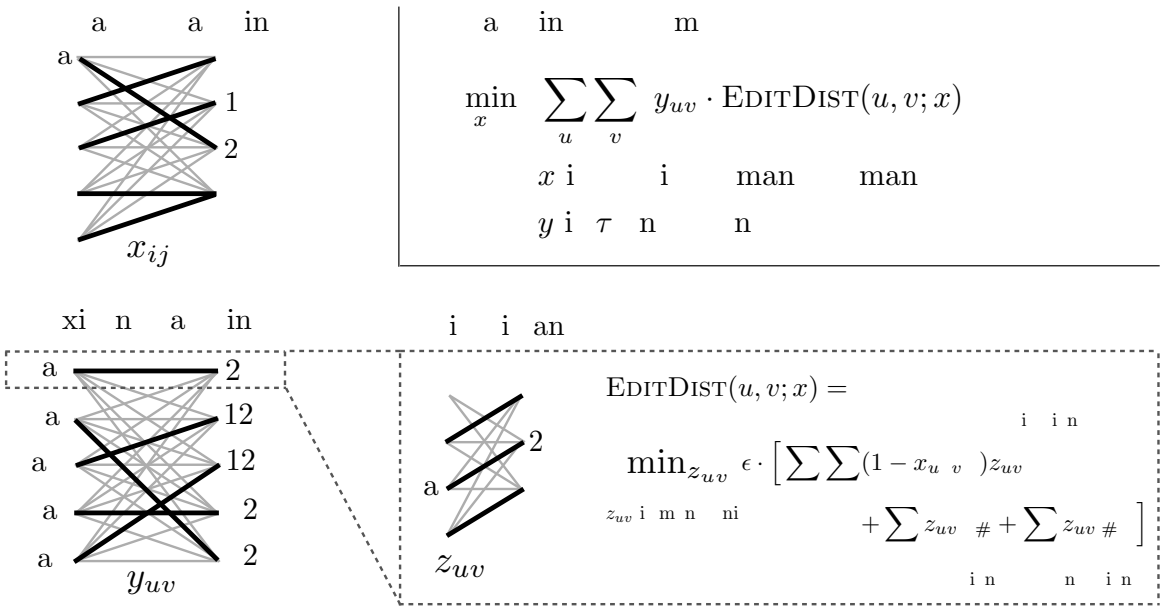

Figure 1: An example problem displaying source and target lexicons and alphabets, along with specific matchings. The variables involved in the optimization problem are diagrammed. x are the alphabet matching indicator variables, y are the lexicon matching indicator variables, and z are the edit alignment indicator variables. The index u refers to a word in the source lexicon, v refers to word in the target lexicon, i refers to a character in the source alphabet, and j refers to a character in the target alphabet. n and m refer to positions in source and target words respectively. The matching objective function is also shown.

v_m. Let z_{uv} be the vector of alignment variables for the edit distance computation between source word u and target word v, where entry $z_{uv,nm}$ indicates whether the character at position n of source word u is aligned to the character at position m of target word v. Additionally, define variables $z_{uv,n\#}$ and $z_{uv,\#m}$ denoting null alignments, which will be used to keep track of insertions and deletions.

$$\text{SUB}(z_{uv}, x) = \sum_{n,m} (1 - x_{u_n v_m}) z_{uv,nm}$$

$$\text{DEL}(z_{uv}) = \sum_n z_{uv,n\#}$$

$$\text{INS}(z_{uv}) = \sum_m z_{uv,\#m}$$

$$\begin{aligned} &\text{EDITDIST}(u, v; x) = \\ &\min_{z_{uv}} \ \epsilon \cdot \Big(\text{SUB}(z_{uv}, x) + \text{DEL}(z_{uv}) + \text{INS}(z_{uv}) \Big) \\ &\text{s.t. } z_{uv} \text{ is monotonic} \end{aligned}$$

We define $\text{SUB}(z_{uv}, x)$ to be the number of substitutions between characters that are not matched in x, $\text{DEL}(z_{uv})$ to be the number of deletions, and $\text{INS}(z_{uv})$ to be the number of insertions.

Notice that the variable $z_{uv,nm}$ being turned on indicates the substitute operation, while a $z_{uv,n\#}$ or $z_{uv,\#m}$ being turned on indicates an insert or delete operation. These variables are digrammed in Figure 1. The requirement that z_{uv} be a monotonic alignment can be expressed using linear constraints, but in our optimization procedure (described in Section 3) these constraints need not be explicitly represented.

Now we can substitute the explicit edit distance equation into the *implicit matching objective* (1).

Noticing that the mins and sums commute, we arrive at the explicit form of the matching optimization problem.

(2) *Explicit Matching Objective*:

$$\min_{x,y,z} \Big[\sum_{u,v} y_{uv} \cdot \epsilon \cdot \big(\textsc{Sub}(z_{uv}, x) + \textsc{Del}(z_{uv}) + \textsc{Ins}(z_{uv}) \big) \Big]$$

s.t. x is restricted-many-to-many

y is τ-one-to-one

$\forall_{uv}\ z_{uv}$ is monotonic

The implicit and explicit optimizations are the same, apart from the fact that the explicit optimization now explicitly represents the edit alignment variables z. Let the *explicit matching objective* (2) be denoted as $J(x, y, z)$. The relaxation of the explicit problem with 0-1 constraints removed has integer solutions,[2] however the objective $J(x, y, z)$ is non-convex. We thus turn to a block coordinate descent method in the next section in order to find local optima.

3 Optimization Method

We now state a block coordinate descent procedure to find local optima of $J(x, y, z)$ under the constraints on x, y, and z. This procedure alternates between updating y and z to their exact joint optima when x is held fixed, and updating x to its exact optimum when y and z are held fixed.

The psuedocode for the procedure is given in Algorithm 1. Note that the function EDITDIST returns both the min edit distance e_{uv} and the argmin edit alignments z_{uv}. Also note that c_{ij} is as defined in Section 3.2.

3.1 Lexicon Matching Update

Let x, the alphabet matching variable, be fixed. We consider the problem of optimizing $J(x, y, z)$ over the lexicon matching variable y and and the edit alignments z under the constraint that y is τ-one-to-one and each z_{uv} is monotonic.

[2]This can be shown by observing that optimizing x when y and z are held fixed yields integer solutions (shown in Section 3.2), and similarly for the optimization of y and z when x is fixed (shown in Section 3.1). Thus, every local optimum with respect to these block coordinate updates has integer solutions. The global optimum must be one of these local optima.

Algorithm 1 Block Coordinate Descent

Randomly initialize alphabet matching x.
repeat
  for all u, v **do**
    $(e_{uv}, z_{uv}) \leftarrow \textsc{EditDist}(u, v; x)$
  end for
  [Hungarian]
  $y \leftarrow \text{argmin}_{y\ \tau\text{-one-to-one}} \Big[\sum_{u,v} y_{uv} e_{uv} \Big]$
  [Solve LP]
  $x \leftarrow \text{argmax}_{x\ \text{restr.-many-to-many}} \Big[\sum_{i,j} x_{ij} c_{ij} \Big]$
until convergence

Notice that y simply picks out which edit distance problems affect the objective. The z_{uv} in each of these edit distance problems can be optimized independently. z_{uv} that do not have y_{uv} active have no effect on the objective, and z_{uv} with y_{uv} active can be optimized using the standard edit distance dynamic program. Thus, in a first step we compute the $U \cdot V$ edit distances e_{uv} and best monotonic alignment variables z_{uv} between all pairs of source and target words using $U \cdot V$ calls to the standard edit distance dynamic program. Altogether, this takes time $O\big((\sum_u l_u) \cdot (\sum_v l_v)\big)$.

Now, in a second step we compute the least weighted τ-one-to-one matching y under the weights e_{uv}. This can be accomplished in time $O(\max(U, V)^3)$ using the Hungarian algorithm (Kuhn, 1955). These two steps produce y and z that exactly achieve the optimum value of $J(x, y, z)$ for the given value of x.

3.2 Alphabet Matching Update

Let y and z, the lexicon matching variables and the edit alignments, be fixed. Now, we find the optimal alphabet matching variables x subject to the constraint that x is restricted-many-to-many.

It makes sense that to optimize $J(x, y, z)$ with respect to x we should prioritize mappings x_{ij} that would mitigate the largest substitution costs in the active edit distance problems. Indeed, with a little algebra it can be shown that solving a maximum weighted matching problem with weights c_{ij} that count potential substitution costs gives the correct update for x. In particular, c_{ij} is the total cost of substitution edits in the active edit alignment prob-

lems that would result if source character i were not mapped to target character j in the alphabet matching x. This can be written as:

$$c_{ij} = \sum_{u,v} \sum_{n,m \text{ s.t. } u_n=i, v_m=j} \epsilon \cdot y_{uv} \cdot z_{uv,nm}$$

If x were constrained to be one-to-one, we could again apply the Hungarian algorithm, this time to find a maximum weighted matching under the weights c_{ij}. Since we have instead allowed restricted-many-to-many alphabet matchings we turn to linear programming for optimizing x. We can state the update problem as the following linear program (LP), which is guaranteed to have integer solutions:

$$\begin{aligned} \min_{x} \quad & \sum_{ij} x_{ij} c_{ij} \\ \text{s.t.} \quad & \forall_i \sum_j x_{ij} \leq 2, \quad \forall_j \sum_i x_{ij} \leq 2 \\ & \sum_i \sum_j x_{ij} = \max(I, J) \end{aligned}$$

In experiments we used the GNU Linear Programming Toolkit (GLPK) to solve the LP and update the alphabet matching x. This update yields matching variables x that achieve the optimum value of $J(x, y, z)$ for fixed y and z.

3.3 Random Restarts

In practice we found that the block coordinate descent procedure can get stuck at poor local optima. To find better optima, we run the coordinate descent procedure multiple times, initialized each time with a random alphabet matching. We choose the local optimum with the best objective value across all initializations. This approach yielded substantial improvements in achieved objective value.

4 Experiments

We compare our system to three different state-of-the-art systems on three different data sets. We set up experiments that allow for as direct a comparison as possible. In some cases it must be pointed out that the past system's goals are different from our own, and we will be comparing in a different way than the respective work was intended. The three systems make use of additional, or slightly different, sources of information.

4.1 Phonetic Cognate Lexicons

The first data set we evaluate on consists of 583 triples of phonetic transcriptions of cognates in Spanish, Portuguese, and Italian. The data set was introduced by Bouchard-Côté et al. (2007). For a given pair of languages the task is to determine the mapping between lexicons that correctly maps each source word to its cognate in the target lexicon. We refer to this task and data set as ROMANCE.

Hall and Klein (2010) presented a state-of-the-art system for the task of cognate identification and evaluated on this data set. Their model explicitly represents parameters for phonetic change between languages and their parents in a phylogenetic tree. They estimate parameters and infer the pairs of cognates present in all three languages jointly, while we consider each pair of languages in turn.

Their model has similarities with our own in that it learns correspondences between the alphabets of pairs of languages. However, their correspondences are probabilistic and implicit while ours are hard and explicit. Their model also differs from our own in a key way. Notice that the phonetic alphabets for the three languages are actually the same. Since phonetic change occurs gradually across languages a helpful prior on the correspondence is to favor the identity. Their model makes use of such a prior. Our model, on the other hand, is unaware of any prior correspondence between alphabets and does not make use of this additional information about phonetic change.

Hall and Klein (2010) also evaluate their model on lexicons that do not have a perfect cognate mapping. This scenario, where not every word in one language has a cognate in another, is more realistic. They produced a data set with this property by pruning words from the ROMANCE data set until only about 75% of the words in each source lexicon have cognates in each target lexicon. We refer to this task and data set as PARTIALROMANCE.

4.2 Lexicons Extracted from Corpora

Next, we evaluate our model on a noisier data set. Here the lexicons in source and target languages are extracted from corpora by taking the top 2,000 words in each corpus. In particular, we used the English and Spanish sides of the Europarl parallel cor-

pus (Koehn, 2005). To make this set up more realistic (though fairly comparable), we insured that the corpora were non-parallel by using the first 50K sentences on the English side and the second 50K sentences on the Spanish side. To generate a gold cognate matching we used the intersected HMM alignment model of Liang et al. (2008) to align the full parallel corpus. From this alignment we extracted a translation lexicon by adding an entry for each word pair with the property that the English word was aligned to the Spanish in over 10% of the alignments involving the English word. To reduce this translation lexicon down to a cognate matching we went through the translation lexicon by hand and removed any pair of words that we judged to not be cognates. The resulting gold matching contains cognate mappings in the English lexicon for 1,026 of the words in the Spanish lexicon. This means that only about 50% of the words in English lexicon have cognates in the Spanish lexicon. We evaluate on this data set by computing precision and recall for the number of English words that are mapped to a correct cognate. We refer to this task and data set as EUROPARL.

On this data set, we compare against the state-of-the-art orthographic system presented in Haghighi et al. (2008). Haghighi et al. (2008) presents several systems that are designed to extract translation lexicons for non-parallel corpora by learning a correspondence between their monolingual lexicons. Since our system specializes in matching cognates and does not take into account additional information from corpus statistics, we compare against the version of their system that only takes into account orthographic features and is thus is best suited for cognate detection. Their system requires a small seed of correct cognate pairs. From this seed the system learns a projection using canonical correlation analysis (CCA) into a canonical feature space that allows feature vectors from source words and target words to be compared. Once in this canonical space, similarity metrics can be computed and words can be matched using a bipartite matching algorithm. The process is iterative, adding cognate pairs to the seed lexicon gradually and each time re-computing a refined projection. Our system makes no use of a seed lexicon whatsoever.

Both our system and the system of Haghighi et al. (2008) must solve bipartite matching problems between the two lexicons. For this data set, the lexicons are large enough that finding the exact solution can be slow. Thus, in all experiments on this data set, we instead use a greedy competitive linking algorithm that runs in time $O(U^2V^2\log(UV))$.

Again, for this dataset it is reasonable to expect that many characters will map to themselves in the best alphabet matching. The alphabets are not identical, but are far from disjoint. Neither our system, nor that of Haghighi et al. (2008) make use of this expectation. As far as both systems are concerned, the alphabets are disjoint.

4.3 Decipherment

Finally, we evaluate our model on a data set where a main goal is to decipher an unknown correspondence between alphabets. We attempt to learn a mapping from the alphabet of the ancient Semitic language Ugaritic to the alphabet of Hebrew, and at the same time learn a matching between Hebrew words in a Hebrew lexicon and their cognates in a Ugaritic lexicon. This task is related to the task attempted by Snyder et al. (2010). The data set consists of a Ugaritic lexicon of 2,214 words, each of which has a Hebrew cognate, the lexicon of their 2,214 Hebrew cognates, and a gold cognate dictionary for evaluation. We refer to this task and data set as UGARITIC.

The non-parameteric Bayesian system of Snyder et al. (2010) assumes that the morphology of Hebrew is known, making use of an inventory of suffixes, prefixes, and stems derived from the words in the Hebrew bible. It attempts to learn a correspondence between the morphology of Ugaritic and that of Hebrew while reconstructing cognates for Ugaritic words. This is a slightly different goal than that of our system, which learns a correspondence between lexicons. Snyder et al. (2010) run their system on a set 7,386 Ugaritic words, the same set that we extracted our 2,214 Ugaritic words with Hebrew cognates from. We evaluate the accuracy of the lexicon matching produced by our system on these 2,214 Ugaritic words, and so do they, measuring the number of correctly reconstructed cognates.

By restricting the source and target lexicons to sets of cognates we have made the task easier. This was necessary, however, because the Ugaritic and Hebrew corpora used by Snyder et al. (2010) are not

Model	τ	Accuracy
Hall and Klein (2010)	–	**90.3**
MATCHER	1.0	90.1

Table 1: Results on ROMANCE data set. Our system is labeled MATCHER. We compare against the phylogenetic cognate detection system of Hall and Klein (2010). We show the pairwise cognate accuracy across all pairs of languages from the following set: Spanish, Portuguese, and Italian.

comparable: only a small proportion of the words in the Ugaritic lexicon have cognates in the lexicon composed of the most frequent Hebrew words.

Here, the alphabets really are disjoint. The symbols in both languages look nothing alike. There is no obvious prior expectation about how the alphabets will be matched. We evaluate against a well-established correspondence between the alphabets of Ugaritic and Hebrew. The Ugaritic alphabet contains 30 characters, the Hebrew alphabet contains 22 characters, and the gold matching contains 33 entries. We evaluate the learned alphabet matching by counting the number of recovered entries from the gold matching.

Due to the size of the source and target lexicons, we again use the greedy competitive linking algorithm in place of the exact Hungarian algorithm in experiments on this data set.

5 Results

We present results on all four datasets ROMANCE, PARTIALROMANCE, EUROPARL, and UGARITIC. On the ROMANCE and PARTIALROMANCE data sets we compare against the numbers published by Hall and Klein (2010). We ran an implementation of the orthographic system presented by Haghighi et al. (2008) on our EUROPARL data set. We compare against the numbers published by Snyder et al. (2010) on the UGARITIC data set. We refer to our system as MATCHER in result tables and discussion.

5.1 ROMANCE

The results of running our system, MATCHER, on the ROMANCE data set are shown in Table 1. We recover 88.9% of the correct cognate mappings on the pair Spanish and Italian, 85.7% on Italian and Portuguese, and 95.6% on Spanish and Portuguese.

Model	τ	Precision	Recall	F1
Hall and Klein (2010)	–	66.9	**82.0**	73.6
MATCHER	0.25	**99.7**	34.0	50.7
	0.50	93.8	60.2	73.3
	0.75	81.1	78.0	**79.5**

Table 2: Results on PARTIALROMANCE data set. Our system is labeled MATCHER. We compare against the phylogenetic cognate detection system of Hall and Klein (2010). We show the pairwise cognate precision, recall, and F1 across all pairs of languages from the following set: Spanish, Portuguese, and Italian. Note that approximately 75% of the source words in each of the source lexicons have cognates in each of the target lexicons.

Our average accuracy across all pairs of languages is 90.1%. The phylogenetic system of Hall and Klein (2010) achieves an average accuracy of 90.3% across all pairs of languages. Our system achieves accuracy comparable to that of the phylogenetic system, despite the fact that the phylogenetic system is substantially more complex and makes use of an informed prior on alphabet correspondences.

The alphabet matching learned by our system is interesting to analyze. For the pairing of Spanish and Portuguese it recovers phonetic correspondences that are well known. Our system learns the correct cognate pairing of Spanish /bino/ to Portuguese /vinu/. This pair exemplifies two common phonetic correspondences for Spanish and Portuguese: the Spanish /o/ often transforms to a /u/ in Portuguese, and Spanish /b/ often transforms to /v/ in Portuguese. Our system, which allows many-to-many alphabet correspondences, correctly identifies the mappings /o/ $\rightarrow$ /u/ and /b/ $\rightarrow$ /v/ as well as the identity mappings /o/ $\rightarrow$ /o/ and /b/ $\rightarrow$ /b/ which are also common.

5.2 PARTIALROMANCE

In Table 2 we present the results of running our system on the PARTIALROMANCE data set. In this data set, only approximately 75% of the source words in each of the source lexicons have cognates in each of the target lexicons. The parameter τ trades off precision and recall. We show results for three different settings of τ: 0.25, 0.5, and 0.75.

Our system achieves an average precision across language pairs of 99.7% at an average recall of 34.0%. For the pairs Italian – Portuguese, and Span-

Model	Seed	τ	Precision	Recall	F1
Haghighi et al. (2008)	20	0.1	72.0	14.0	23.5
	20	0.25	63.6	31.0	41.7
	20	0.5	44.8	43.7	44.2
	50	0.1	90.5	17.6	29.5
	50	0.25	75.4	36.7	49.4
	50	0.5	56.4	55.0	55.7
MATCHER	0	0.1	**93.5**	18.2	30.5
	0	0.25	83.2	40.5	54.5
	0	0.5	56.5	**55.1**	**55.8**

Table 3: Results on EUROPARL data set. Our system is labeled MATCHER. We compare against the bilingual lexicon induction system of Haghighi et al. (2008). We show the cognate precision, recall, and F1 for the pair of languages English and Spanish using lexicons extracted from corpora. Note that approximately 50% of the words in the English lexicon have cognates in the Spanish lexicon.

ish – Portuguese, our system achieves prefect precision at recalls of 32.2% and 38.1% respectively. The best average F1 achieved by our system is 79.5%, which surpasses the average F1 of 73.6 achieved by the phylogenetic system of Hall and Klein (2010).

The phylogenetic system observes the phylogenetic tree of ancestry for the three languages and explicitly models cognate evolution and survival in a 'survival' tree. One might expect the phylogenetic system to achieve better results on this data set where part of the task is identifying which words do not have cognates. It is surprising that our model does so well given its simplicity.

5.3 EUROPARL

Table 3 presents results for our system on the EUROPARL data set across three different settings of τ: 0.1, 0.25, and 0.5. We compare against the orthographic system presented by Haghighi et al. (2008), across the same three settings of τ, and with two different sizes of seed lexicon: 20 and 50. In this data set, only approximately 50% of the source words have cognates in the target lexicon.

Our system achieves a precision of 93.5% at a recall of 18.2%, and a best F1 of 55.0%. Using a seed matching of 50 word pairs, the orthographic system of Haghighi et al. (2008) achieves a best F1 of 55.7%. Using a seed matching of 20 word pairs, it achieves a best F1 of 44.2%. Our system outperforms the orthographic system even though the orthographic system makes use of important additional information: a seed matching of correct cognate pairs. The results show that as the size of this seed is decreased, the performance of the orthographic system degrades.

Model	τ	Lexicon Acc.	Alphabet Acc.
Snyder et al. (2010)	–	60.4*	**29/33***
MATCHER	1.0	**90.4**	28/33

Table 4: Results on UGARITIC data set. Our system is labeled MATCHER. We compare against the decipherment system of Snyder et al. (2010). *Note that results for this system are on a somewhat different task. In particular, the MATCHER system assumes the inventories of cognates in both Hebrew and Ugaritic are known, while the system of Snyder et al. (2010) *reconstructs* cognates assuming only that the morphology of Hebrew is known, which is a harder task. We show cognate pair identification accuracy and alphabet matching accuracy for Ugaritic and Hebrew.

5.4 UGARITIC

In Table 4 we present results on the UGARITIC data set. We evaluate both accuracy of the lexicon matching learned by our system, and the accuracy of the alphabet matching. Our system achieves a lexicon accuracy of 90.4% while correctly identifying 28 out the 33 gold character mappings.

We also present the results for the decipherment model of Snyder et al. (2010) in Table 4. Note that while the evaluation data sets for our two models are the same, the tasks are very different. In particular, our system assumes the inventories of cognates in both Hebrew and Ugaritic are known, while the system of Snyder et al. (2010) reconstructs cognates assuming only that the morphology of Hebrew is known, which is a harder task. Even so, the results show that our system is effective at decipherment when semantically similar lexicons are available.

6 Conclusion

We have presented a simple combinatorial model that simultaneously incorporates both a matching between alphabets and a matching between lexicons. Our system is effective at both the tasks of cognate identification and alphabet decipherment, requiring only lists of words in both languages as input.

References

A. Bouchard-Côté, P. Liang, T.L. Griffiths, and D. Klein. 2007. A probabilistic approach to diachronic phonology. In *Proc. of EMNLP*.

A. Bouchard-Côté, T.L. Griffiths, and D. Klein. 2009. Improved reconstruction of protolanguage word forms. In *Proc. of NAACL*.

A. Haghighi, P. Liang, T. Berg-Kirkpatrick, and D. Klein. 2008. Learning bilingual lexicons from monolingual corpora. *Proceedings of ACL*.

D. Hall and D. Klein. 2010. Finding cognate groups using phylogenies. In *Proc. of ACL*.

K. Knight and K. Yamada. 1999. A computational approach to deciphering unknown scripts. In *Proc. of ACL Workshop on Unsupervised Learning in Natural Language Processing*.

K. Knight, A. Nair, N. Rathod, and K. Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proc. of COLING/ACL*.

P. Koehn and K. Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proc. of ACL workshop on Unsupervised lexical acquisition*.

P. Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proc. of Machine Translation Summit*.

G. Kondrak. 2001. Identifying Cognates by Phonetic and Semantic Similarity. In *NAACL*.

H.W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*.

P. Liang, D. Klein, and M.I. Jordan. 2008. Agreement-based learning. *Proc. of NIPS*.

S. Ravi and K. Knight. 2011. Bayesian inference for Zodiac and other homophonic ciphers. In *Proc. of ACL*.

B. Snyder, R. Barzilay, and K. Knight. 2010. A statistical model for lost language decipherment. In *Proc. of ACL*.