Automatic Extraction of Morphological Lexicons from Morphologically Annotated Corpora

Ramy Eskander, Nizar Habash, Owen Rambow
Center for Computational Learning Systems
Columbia University
{reskander,habash,rambow}@ccls.columbia.edu

Abstract

We present a method for automatically learning inflectional classes and associated lemmas from morphologically annotated corpora. The method consists of a core language-independent algorithm, which can be optimized for specific languages. The method is demonstrated on Egyptian Arabic and German, two morphologically rich languages. Our best method for Egyptian Arabic provides an error reduction of 55.6% over a simple baseline; our best method for German achieves a 66.7% error reduction.

1 Introduction

Morphological lexicons specify all inflected forms for each lexeme; in a language with rich morphology, such a resource can be important for natural language processing (NLP) tasks in order to limit data sparseness. For example, a morphological lexicon is an important component of a morphological tagger or of a part-of-speech (POS) tagger for languages with rich morphology.[1] Traditionally, a morphological lexicon has been created through painstaking lexicographic and morphological analysis of the language, drawing on unannotated corpora. Recently, new approaches have emerged. Fully or largely unsupervised approaches cannot link surface forms to morphosyntactic features and are thus not suited for building morphological lexicons. This problem is overcome by approaches that use explicit linguistic knowledge. In this paper, we investigate using existing morphologically annotated corpora. In a morphologically annotated corpus, the words in naturally occurring texts or transcribed speech are annotated for the correct morphological analysis (including, of course, core POS) in context. While there has been much work on computational morphology, to our knowledge this is the first paper to study the question of how to extract morphological lexicons from morphologically annotated corpora, and how to determine how much annotation is needed. In this paper, we assume a corpus with each word annotated with morphosyntactic features and with a lemma which tells us what lexeme the word form is part of. The task is to predict the correspondence between a word form and its lemma and morphological features.

This paper makes two contributions. First, we introduce an algorithm that learns unseen forms by analogy. This algorithm is language-independent. It incrementally merges complementary paradigm information about different lexemes into more abstract and more informative inflectional classes. Second, we explore how to model stems, and we propose a generalization of the Semitic root-and-template modeling. We use Egyptian Arabic (EGY), and German (GER) as our test languages. We test on corpus data, in order to simulate a standard real-world application. The baseline just uses the word forms seen in training and does not predict any unseen forms. Our language-independent algorithm improves the performance for both EGY and GER, with error reductions over the baseline of 44.4% for EGY and of 66.7% for GER. By adding language-specific modeling of the stem using templates, we obtain further error reductions for EGY (up to 55.6%) but not for GER.

Next, we review related work (Section 2) and introduce the key linguistic concepts we use (Section 3). We present our basic language-independent method in Section 4, and our language-specific modeling of stem variation in Section 5.

[1] Note that a full-form morphological lexicon is functionally equivalent to a morphological analyzer-generator, since one can be used to create the other.

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1032–1043,
Seattle, Washington, USA, 18-21 October 2013.

2 Related Work

Approaches to Morphological Modeling Much work has been done in the area of computational morphology ranging from systems painstakingly designed by hand (Koskenniemi, 1983; Buckwalter, 2004; Habash and Rambow, 2006; Détrez and Ranta, 2012) to unsupervised methods that learn morphology models from unannotated data (Creutz and Lagus, 2007; Monson et al., 2008; Hammarström and Borin, 2011; Dreyer and Eisner, 2011). There is a large continuum between these two approaches. Closer to one end, we find work on minimally supervised methods for morphology learning that make use of available resources such as parallel data, dictionaries or some additional morphological annotations (Yarowsky and Wicentowski, 2000; Cucerzan and Yarowsky, 2002; Neuvel and Fulop, 2002; Snyder and Barzilay, 2008). Closer to the other end, we find work that focuses on defining morphological models with limited lexicons that are then extended using raw text (Clément et al., 2004; Forsberg et al., 2006). The work presented in this paper falls in the middle of this continuum: we are interested in learning complete morphological models using rich morphological annotations and, optionally, limited linguistic knowledge. We compare the value of different amounts of annotation and how they relate to additional linguistic knowledge.

Morphological Paradigms Many traditional and modern theories of inflectional morphology organize natural language morphology by paradigms (Stump, 2001; Walther, 2011; Camilleri, 2011). Within the continuum we discussed above, we find hierarchical representations of paradigm knowledge that have been used in manually constructed morphological models (Finkel and Stump, 2002; Habash et al., 2005). Furthermore, Détrez and Ranta (2012) introduce an implementation of Smart Paradigms – heuristically organized paradigms minimizing the number of forms needed to predict the full paradigm of a particular lexeme.

Both Forsberg et al. (2006) and Clément et al. (2004) describe methods for automatically populating a lexicon from raw data given a set of morphological inflectional classes in a language. Our work differs in that we use annotated data, but do not start with a complete set of inflectional classes; thus, our work is exactly complementary to this work.

The concept of a paradigm is also used in many published efforts on unsupervised learning of morphology, although not always in a way consistent with its use in linguistics. For instance, Snover et al. (2002) (and later on Can and Manandhar (2012)) define a paradigm as "a set of suffixes and the stems that attach to those suffixes and no others". This definition is quite limited since it is not modeling the notion of lexeme. Chan (2006) defines a simpler concept of paradigms in his *probabilistic paradigm* model, which has many limitations, such as not handling syncretism or irregular morphology, nor distinguishing inflection and derivation.

Dreyer and Eisner (2011) learn complete German verb paradigms from a small set of complete seed paradigms (50 or 100), which they choose randomly from all verbs in the language. They model stem changes using letter-based models, and use a large unannotated corpus in addition to the seed paradigms. Durrett and DeNero (2013) attack the same problem as Dreyer and Eisner (2011). Instead of using unannotated text, they model explicit rules for affixes and stem changes. The major difference between these two efforts and our work is that the problem is defined differently: we assume that the training and test data is defined by a corpus, not by complete paradigms. Our methods therefore are more sensitive to frequency effects of tokens. We believe that our way of stating the problem is more relevant to actual computational challenges for languages with limited morphological resources. We empirically compare our approach to that of Durrett and DeNero (2013) in Section 5.

None of the unsupervised approaches mentioned model inflectional classes, i.e., meta-paradigmatic representation that cluster the various paradigms of different lexemes into a set of general classes of paradigms. There is no explicit notion of morphosyntactic features. In this paper we target the learning and completion of inflectional classes from morphologically annotated data. Our approach does not sacrifice details of what paradigms should include: we handle syncretism and stem changes, and allow for the prediction of new word forms from morphosyntactic features and lemmas, unlike the largely unsupervised work. Our work also differs from most previous work in that we investigate how to model stem change explicitly. Whereas other approaches model stem syncretism through letter-

based models (Yarowsky and Wicentowski, 2000; Neuvel and Fulop, 2002; Dreyer and Eisner, 2011), we explore the use of abstract stems.

In our previous work on the EGY morphological analyzer CALIMA, we similarly used a lexicon of annotated morphological forms and extended it automatically using a simpler approach to paradigm completion (Habash et al., 2012).

3 Linguistic Terminology

In this section, we review key concepts from morphology, and introduce the terminology we will use in this paper.[2] We then introduce our own formalization of stems using vocalic templates.

Morphology is the study of word forms and their decomposition into elementary morphemes, which are the smallest meaning-bearing units of a language. There are two types of morphological processes: inflectional and derivational morphology. In inflectional morphology, a core meaning is retained and different word forms reflect different types of morphosyntactic features such as person, number, or tense. In derivational morphology, the core meaning of a word is changed, and perhaps even its part-of-speech (POS). In this paper, we restrict our interest to inflectional morphology. Furthermore, we take the written form of the word to be primary, and base all morphological analyses on the written form.

We will refer to the set of all word forms that are related through inflectional morphology alone as **lexeme**. We can refer to a lexeme with a **lemma**, which we take to be a language-specific and conventionalized choice of one of the inflected forms. For example, in English the verb is conventionally cited in the infinitive (often with *to*, which can be omitted), while in Arabic it is conventionally cited in perfective third person masculine singular. The lemma is sometimes referred to as a "citation form". A **paradigm** of a lexeme is a list of **cells**, where a cell is a combination of a complete set of morphosyntactic features (properties) and the corresponding inflected form of the lexeme. A paradigm is **complete** if there are cells for all possible morphosyntactic features (the list of possible morphosyntactic features is of course language-dependent).

We can divide the word forms into **affixes** (i.e., prefixes and suffixes) and the **stem**. There is no single correct way to do this for the words of a language. Each lexeme has its own paradigm. We can abstract from paradigms by grouping together paradigms which share the same affixes in corresponding cells, and where stems in corresponding cells differ in some restricted manner. We can define the **inflectional class** (IC) more formally as a set of **abstract cells**, where an abstract cell is a combination of a complete set of morphosyntactic features (properties) and an abstract representation of a stem (an **abstract stem**) along with fully specified affixes. The abstract stems (and thus the abstract cells) must have the property that, given a single instantiated word form of a lexeme along with its associated IC, we can derive the *complete* paradigm of the lexeme deterministically. In the first results we present, we simply assume that the stem is either shared entirely with other abstract cells, or it is entirely lexically instantiated. We explore language-specific approaches to defining an abstract stem in Section 5, where we also discuss relevant morphological facts of EGY and GER.

Prefixes, suffixes, and stems can be the same for different cells of a single paradigm or IC. This is called **syncretism**. We will refer to the cells which share a stem as a **stem syncretism zone** or "zone" for short.

4 Language-Independent Inflectional Class Construction Algorithm

In this section, we present our language-independent IC construction algorithm (LICA). LICA consists of a core algorithm for building ICs from seen data and models of soft-stem syncretism and affix prediction.

4.1 Problem Definition

Starting with a corpus of words annotated as triples of ⟨*prefix*+*stem*+*suffix*, *lemma*, *features*⟩, we want to create a lexicon of complete ICs, with each IC having an associated set of lemmas. The following is an input example specifying the inflected form of the 3rd person plural imperfective inflection for the EGY lemma **katab**[3] 'write': ⟨y+iktib+uwA, **katab**, I3UP⟩.

[2] We (roughly) base our terminology and our conceptualization on the inferential-realizational theory of Stump (2001).

[3] Arabic transliteration throughout the paper is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007).

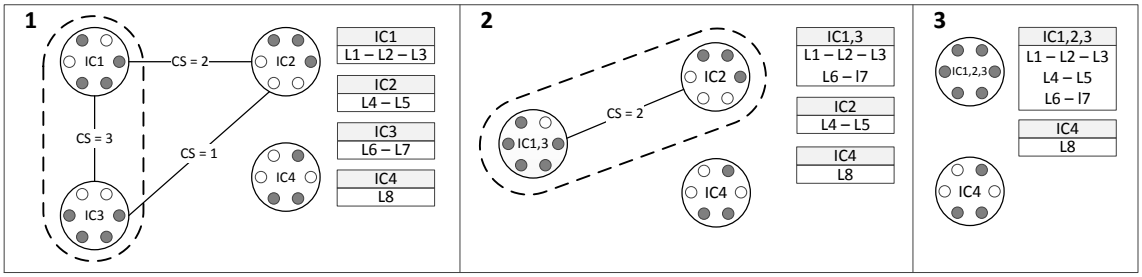

Figure 1: This graph illustrates two merges that result in combining three ICs in two steps. The IC cells are represented as solid (filled cells) or blank (empty cells) circles. CS is the compatibility score marking the number of matching filled cells in an IC. The boxes to the right of the graph represent the lexicon which associates lemmas (L*) with ICs. IC4 is not connected with any of the other ICs, because it has cell values that are incompatible with them.

4.2 Core Algorithm

Building the Initial Inflectional Classes An initial IC is constructed for each lemma found in the input corpus using all the triplets involving said lemma. Since the lemma itself is an inflected form with an *a priori* fixed feature combination, we also use the lemma to construct the initial IC, even if it did not occur as a word form in the corpus. If all inflected forms of a lemma appear in the training data, the IC will be complete. However, typically there are many unseen forms. If all seen forms related to one lexeme have the same stem, the abstract stem chosen for the IC is a single stem variable; the abstract cells of the new IC can of course differ in terms of affixes (which are always fully specified rather than represented as variables). In cases when the seen forms of one lemma have more than one stem, the IC created will simply be the same as the paradigm, i.e., we have fully instantiated stems as our abstract stems. We call such ICs **suppletive ICs**. At this point, we have a repository of numerous incomplete ICs and a lexicon consisting of a one-to-one mapping between lemmas and these new ICs. We then identify all ICs that are exactly the same (by definition, these are not suppletive ICs), and merge the associated sets of lemmas. We now have a new lexicon in which some ICs are associated with more than one lemma.

Constructing the Inflectional Class Graph We construct an IC graph that connects all *mergeable* ICs. Two ICs are mergeable if neither of the ICs is suppletive, if they share at least one abstract cell for some morphological feature, and if no morphological features are associated with different abstract cells, i.e., there are no incompatibilities. Each point in the graph represents a specific IC, while the edges carry the following four scores that we use to determine mergeability order:

- The **compatibility score** is the number of non-empty intersections between the two ICs.
- The **originality score** is the larger number of previous merges of the two ICs. At the beginning, this number is 0 for all edges.
- The **completeness score** is the size of the union of non-empty rows from the two ICs
- The **lexical size score** is the sum of the number of lemmas associated with the two ICs.

The edges of the graph are ranked according to the compatibility score first (more is better). Any ties are broken using the originality score (lower is better), any remaining ties by the completeness score (more is better) and then by the lexical size score (more is better). We explored all possible orders of tie breaking using EGY data, and determined the order of the listing above to be best. We use it in all experiments reported in this paper.

Merging the Inflectional Classes While the IC graph is still connected, we repeat the following merging procedure. Starting with the highest ranked edge in the graph, we merge the two ICs connected by the edge. In case of multiple edges that are equally highly ranked (after tie breaking), we select randomly among them. The merge creates a new IC that has the union of the cells in the two ICs. The lexicon is adjusted accordingly by associating with the new IC the union of all the lemmas originally associated with the two ICs. The new IC inherits the union of all the IC graph connections of its predecessors. The IC graph edge scores between the new

IC and other ICs are recalculated. Graph edges that become incompatible after the merge are removed. The two original ICs are removed from the graph and lexicon. See Figure 1 for an illustration of the merge process.

4.3 Completing Stems and Affixes

Soft Stem Syncretism Zones We extend the concept of stem-syncretism zones into a statistical model that computes the probability that the abstract stems in two abstract cells are the same given their feature combinations, i.e., that they belong to the same syncretism zone. We refer to this approach as "soft" stem-syncretism zones or "soft zones" for short. The probabilities are computed for each feature-combination pair as the ratio of the times the abstract stem for the feature-combination pair are equal divided by the times the abstract stem for the feature-combination pair are not empty. The probabilities can be computed after the initial IC graph construction or after the merging process has concluded; they can also be based on IC type counts or weighted by the number of associated lemmas. Applying soft zones to fully complete the ICs can only be done after merging is completed. We try all the possible alternatives for learning the SZs on the EGY data, including experiments with small training data sizes. The best accuracy is obtained using IC type weights learned after the merge. We use this setting for all experiments.

When applying the soft zones to determine the abstract stems of empty cells, we consider all filled cells in the same IC and select the abstract stem from the cell of the feature combination that has the highest soft-zone probability with the feature combination of the empty cell. Note that the copied abstract stem can be either a stem variable, or, for a suppletive IC, a lexical form.

Predicting Affixes In building the ICs mentioned above, some feature combinations and their corresponding affixes (prefixes and suffixes) are missing since they were not in the training data. We fill the missing affixes in a particular IC i by copying them from other ICs that we rank by overall seen affix similarity to IC i. ICs with conflicting affixes for any feature combinations are excluded completely. For any remaining missing affix-feature combination, we use the most common affix for that feature combination over all ICs.

4.4 Model Application

The complete ICs produced by the completion algorithm, associated with their lemmas in the lexicon, can now be used to predict the surface form of a given lemma and morphological features. We use the following procedure: If the given features are the same as those used to define the lemma, then the surface form is the same as the lemma form. If the lemma was seen in training, we select its IC from the model and generate the inflection associated with the features. This may require instantiating a stem variable (in case the IC is not suppletive), but this can be done deterministically from the lemma.

If the lemma has not been seen, then we build an IC on the fly using the lemma, i.e., an IC with a single cell. We then pick an IC from the model that does not conflict with that IC. The priority is given to the IC with the largest lexicon size, which is likely to be a result of several merges. If such an IC does not exist, a backup mode returns the stem of the lemma associated with the most frequent affixes of the queried features.

4.5 Results on Egyptian Arabic

Data and Metrics We use a morphologically annotated EGY corpus based on the CALLHOME EGY (CHE) corpus (Gadalla et al., 1997).[4] We divide the corpus into three parts: training, development and test, of about 75K, 36K and 41K words, respectively. We conduct our experiments on verbs only since they have a large number of possible morphosyntactic feature combinations. The verbs in these experiments are uncliticized. Clitics are easily handled using a few orthographic rules (El Kholy and Habash, 2010). On average, uncliticized verbs are about 12% of all words in our corpus. The data is represented in triplets as described in Section 4.1. There is a total of 19 feature combinations of aspect/mood (perfective,imperfective and imperative), person (first, second and third), gender (masculine, feminine and neutral) and number (singular and plural). Some combinations are invalid such as the first and third persons with the imperative form. The lemma we use is the Arabic citation form for verbs, which is the perfective third person masculine sin-

[4]The corpus was automatically annotated using information from the CHE transcripts (Gadalla et al., 1997) and the Egyptian Colloquial Arabic Lexicon (Kilany et al., 2002). For more details, see Habash et al. (2012) and Eskander et al. (2013).

gular (P3MS) inflection. We use this fact to fill the P3MS cells when building the initial ICs.

We evaluate the accuracy of the automatically generated bidirectional lexicon by generating surface forms from lemmas and morphosyntactic features. We use the model application method described in Section 4.4.

Baseline Our baseline system consists of two steps. First, we check the features. All cases of citation form features (in the case of EGY, P3MS) return the lemma as the inflected form. Otherwise, we look up the lemma and feature pair among all triplets in the training data and return the inflected form if such a triplet was found. If not, the baseline does not return an answer.

Results The results on tokens are summarized in Table 2 under the column heading BL (baseline) and NOTMP (LICA with no stem template). Our system consistently improves over the baseline for all training data sizes explored.

Error Analysis We performed an error analysis using the best settings found on the development set. We found that in 46% of the error types the lemma is unseen. Additional 35% of the cases are due to stem templates that are unseen in the complete ICs although their corresponding lemmas are seen. About one tenth of the cases are because of the existence of multiple forms of the same lemma and feature combinations, where our system assigns a form that is different from the gold form. Finally, gold errors contribute to about 9% of all errors.

4.6 Results on German

Data and Metrics For our experiments, we use the TIGER corpus (Brants and Hansen, 2002). We divide the corpus into three parts: training, development and test, of about 709K, 143K and 37K words, respectively. However, we use the first 75K words in training and the first 36K words in development to have the results comparable to EGY. We conduct our experiments on verbs only. We disregard any verbs with separable prefixes (as is common in work in morphology learning). On average, verbs without separable prefixes are about 9% of all words in our corpus. The data is represented in triplets as described in Section 4.1. There are 28 feature combinations of tense (present, past), person (first, second and third), number (singular, plural), and mood (indicative, subjunctive, imperative, past participle, infinitive). Some combinations are invalid such as any tense with the non-tensed participle or infinitive. The infinitive is the lemma. We use the same metrics as for EGY (Section 4.5).

Results The results are summarized in Table 5, with the relevant results in the column NOTMP. We see that our algorithm performs substantially better than the baseline at all training set sizes, with greater relative error reductions at smaller training sizes.

Error Analysis We inspected all error types in the development set and found that nearly 8% of all error types are errors in the gold annotation of the development set, and in 4% of cases, our predicted form is correct because a lemma is shared by two verbal paradigms (for example, *werden* has different past participles depending on whether it is the passive auxiliary or the verb for 'become').

5 Language-Specific Modeling of Stems

5.1 General Approach

We model the abstract stems using the notions of **orthographic template** and **orthographic root**. To meet our definition of IC given above, we define these two notions so that the root and template can always be extracted deterministically. We define them simply in terms of sets of letters: the set of letters of the alphabet used to write the language we are modeling is partitioned into the **root letters** and the **pattern letters**. The orthographic template is a string that specifies the template letters and that has placeholders for the root letters, which we write as '□'. The orthographic root is a sequence of strings that specifies the root letters that can fill a vocalic template's placeholders, in the order specified. Note that an IC along with a root is equivalent to a paradigm, since the root can be inserted deterministically into the abstract stem. This gives us another way to specify a lexeme: since a complete paradigm enumerates all inflected forms of a lexeme, and since a complete IC along with a root defines a complete paradigm, we can specify a lexeme to be a pair consisting of an IC along with a root. This pair determines a complete mapping from morphosyntactic features to surface word forms for the lexeme.

The basic algorithm discussed earlier is modified as follows: when we read in the training data, the

EGY Inflectional Class (IC) Repository			
	IC-1	IC-2	IC-3
P1US	□a□a□+t	□a□∼+ayt	□u□+t
P1UP	□a□a□+nA	□a□∼+aynA	□u□+nA
P2MS	□a□a□+t	□a□∼+ayt	□u□+t
P2FS	□a□a□+tiy	□a□∼+aytiy	□u□+tiy
P2UP	□a□a□+tuwA	□a□∼+aytuwA	□u□+tuwA
P3MS	□a□a□	□a□∼	□A□
P3FS	□a□a□+it	□a□∼+it	□A□+it
P3UP	□a□a□+uwA	□a□∼+uwA	□A□+uwA
I1US	Aa+□□i□	Aa+□i□∼	Aa+□uw□
I1UP	ni+□□i□	ni+□i□∼	ni+□uw□
I2MS	ti+□□i□	ti+□i□∼	ti+□uw□
I2FS	ti+□□i□+iy	ti+□i□∼+iy	ti+□uw□+iy
I2UP	ti+□□i□+uwA	ti+□i□∼+uwA	ti+□uw□+uwA
I3MS	yi+□□i□	yi+□i□∼	yi+□uw□
I3FS	ti+□□i□	ti+□i□∼	ti+□uw□
I3UP	yi+□□i□+uwA	yi+□i□∼+uwA	yi+□uw□+uwA
C2MS	A i□□i□	□i□∼	□uw□
C2FS	A i□□i□+iy	□i□∼+iy	□uw□+iy
C2UP	A i□□i□+uwA	□i□∼+uwA	□uw□+uwA

EGY Lexicon					
IC-1		IC-2		IC-3	
katab	**faraD**	**mad∼**	**Hal∼**	**šAf**	**qAl**
‖k,t,b‖	‖f,r,D‖	‖m,d‖	‖H,l‖	‖š,f‖	‖q,l‖
‘write’	‘suppose’	‘extend’	‘solve’	‘see’	‘say’

Table 1: Example of three inflectional classes and associated lemmas with their roots in EGY. The various blocks in the IC table specify different syncretism zones.

root of the lemma is used to decompose the stem of the inflected form into a root and a template.

5.2 Choosing Root and Pattern Letters

Determining which letters count as root letters and which count as pattern letters is language-specific. In this paper, we use three approaches to choosing root and pattern letters.

No Template (NOTMP) In this approach, we do not model the stem at all, i.e., there are no pattern letters. As a result, the ICs cannot express stem changes; any verb with a stem change will be an irregular verb and it will be modeled with a suppletive IC. Note that in EGY, *every* verb manifests a stem change between the perfective and imperfective forms, while in GER many verbs in fact do have the same stem for all inflected forms. The results of the NOTMP approach are the same basic approach results already presented in Section 4.

Scholar (SCHLR) In this approach, we use linguistic scholarship and intuition to choose a set of letters to be the pattern letters. These are letters which we know can change in stems within the same IC. We discuss this approach for EGY in Section 5.3.1 and for GER Section 5.4.1.

Empirical (EMPR) In this approach, we obtain the set of pattern letters empirically. We align the letters of all the stems in the development corpus with the stem letters of their corresponding lemmas. This allows us to identify which letters change between lemma stem and inflected stem. We order the letters by the probability that a letter changes given all occurrences of that letter, and by the probability that a letter changes given all changes that occur. This gives us two rankings. For both rankings and for each letter that changes, we construct a set of letters by including all letters from the most highly ranked letter down to the letter under consideration. This gives us per ranking as many sets as there are letters that change. We then choose the best performing set among all sets generated by both rankings. We present the results of this approach for EGY in Section 5.3.1 and for GER in Section 5.4.1.

5.3 Egyptian Arabic

5.3.1 Choice of Pattern Letters

Scholar-based Pattern Letters For the EGY SCHLR approach, we selected the following pattern letters: اأإآوؤيىئءـُـِـّ *AÂĂĀwŵyýŷ’aui∼*. These letters cover all the so-called weak root radicals, Hamzated forms and diacritics that are often used to discuss different verbal paradigms in Arabic (Gadalla, 2000). Table 1 shows three EGY inflectional classes. We follow the standard practice in modeling Arabic morphology of assuming that each placeholder in an orthographic template corresponds to exactly one letter from the orthographic root (i.e., the root is a sequence of strings each of length one). We would like to stress that our orthographic template and root differ from the notions of pattern and root in Semitic morphology (for an overview, see (Habash, 2010)): while for **katab** ‘write’ the orthographic root as well as the “real” root is ‖k,t,b‖, the orthographic root for **xal∼aý** ‘let’ is ‖x,l‖ (as opposed to the real root ‖x,l,y‖). Henceforth, we will omit the adjective “orthographic”, since we will not talk about the “real” root. To inflect a lemma for a particular set of features, we combine the root with the □ slots in the corresponding inflectional class feature row, e.g., Hal∼ + P2FS ⇒ ‖H,l‖ +

□a□∼+aytiy ⇒ Hal∼aytiy. Note that in this paper, we do not use any rules; all regular phonological and orthographic variation is "compiled" into the ICs. We also see examples of stem syncretism zones in Table 1; they are marked with horizontal lines. The stems associated with P3MS, P3FS and P3UP happen to be the same within each IC (although different across ICs). Different ICs have different zones: e.g., IC-1 has one zone for the P*** features, while IC-2 and IC-3 have two identical zones for P3** and P[12]**.

Empirically Determined Pattern Letters For EGY, the EMPR approach, using the algorithm described in Section 5.2, yields the following optimal set of pattern letters: اأإويىيئءثـّـُـِ *AÂĂwyýŷ'θaui*. This set is very similar to the SCHLR set except in that it omits the lexically constant (per lexeme) Shadda diacrtic ّ ∼ (orthographic gemination marker), and some very infrequent Hamzated forms; and it also adds one letter ث θ as a result of orthographic inconsistency in the training data.

Corpus Size	Verb Count	BL	NOTMP	SCHLR	EMPR	EMPR IIC+HZ
0K	0	19.3	20.0	20.0	20.0	58.8
1K	95	51.8	64.2	68.3	68.1	83.3
5K	630	64.2	77.0	83.9	83.9	91.1
10K	1,201	70.4	84.3	89.4	89.1	92.4
25K	2,994	79.6	91.5	94.4	94.8	95.6
50K	5,966	84.5	94.2	96.9	96.7	97.2
75K	8,690	85.6	94.9	97.5	97.6	97.2

Table 2: Learning curve comparing system performance for EGY on **tokens** (development set).

Results The template approaches SCHLR and EMPR consistently beat NOTMP which does not make use of any templatic stem modeling (see Table 2). However, SCHLR and EMPR perform about equally. This is not surprising since the two sets of pattern letters are quite similar.

Error Analysis We conducted an error analysis of EMPR using the best settings found on the development set. About 86% of the error types in NOTMP where the lemma is unseen are solved after introducing EMPR. Additionally, 71% of the cases in NOTMP where the stem template is unseen and the lemma is seen are solved. However, 7% of the error types in EMPR are not present in NOTMP, and they are all cases where the stem template is unseen while the lemma is seen. Errors due to multiple stem forms and gold errors remain the same in both NOTMP and EMPR, contributing to 26% and 23%, respectively, of all the error types in EMPR.

5.3.2 Other Enhancements with Linguistic Knowledge

Other linguistic knowledge can also help enrich the process of IC learning, especially under limited annotation conditions. We use the following two types of linguistic knowledge, which seamlessly integrate into the core merging algorithm described above.

Iconic Inflectional Classes (IICs) are ICs that are manually fully annotated, i.e., they have all the template cells for all morphosyntactic features specified. IICs are treated like any other ICs when constructing the initial IC graph. They are different from other ICs in that they initially have no lemmas associated with them in the lexicon.

Hard Stem-Syncretism Zones (HZ) are stem-syncretism zones determined manually by linguists to hold for all ICs. As such they can be more fine-grained than is needed to describe individual ICs. A hard zone is not applied in case of any partial disagreement within it. Unlike soft zones, they could be applied before or after the merge process, and they do not guarantee that the ICs will be completely filled.

We conducted experiments where we added external linguistic knowledge to our training data. We added 112 IICs which are extracted from all EGY verb inflections listed in a reference grammar of EGY (Gadalla, 2000). Also we added the following eight HZs for EGY (with reference to features $<tense, person, gender, number>$): (P1US-P1UP-P2MS-P2FS-P2UP), (I1UP-I2MS-I3MS-I3FS), (I2FS-I2UP-I3UP), (P3FS-P3UP), (C2FS-C2UP), (P3MS), (I1US), and (C2MS). Applying the HZs on the completed ICs (before soft zone application) gives higher results than applying them on initial ICs. We only report below on the setting of applying HZs after IC merge completion. We present the accuracies of IC learning for the baseline, and for the best setup with and without IICs and HZs, for different training sizes in Table 2. The baseline with no training data is at 19.3%

because of all the cases with verbs appearing in the citation form. As expected, IICs always help improve accuracy, especially under limited (and no) data conditions. However the benefits diminish rapidly for larger training sets. When evaluating on types only (results not presented in this paper), we find that using IICs in our system with no data is better than using the baseline with 75K words.

5.3.3 Blind Test Set

Table 3 shows the accuracies the different systems on our blind test set. The results for the test set are lower than those of the development set, but the trends are the same. We also compare our results to those obtained using the system of Durrett and DeNero (2013) on the same test data. Note that we apply their system to our problem – predicting unseen forms from annotated corpora (i.e., incomplete paradigms), not to the problem for which they created their system – predicting unseen forms from complete paradigms. Our best system outperforms theirs by 2.8% absolute in accuracy.

System	Accuracy	Error Reduction
Baseline	84.7	
NOTMP	91.5	44.4
SCHLR	93.2	55.6
EMPR	93.2	55.6
Durrett & DeNero	90.4	37.3

Table 3: Results for EGY on **tokens** using a blind test set.

5.4 German

5.4.1 Choice of Pattern Letters

Scholar-based Pattern Letters We now discuss our scholarship-based choice of pattern letters for German. Like Arabic, German verb paradigms can show stem changes which are typically vowel changes. Furthermore, like Arabic, German has prefixes, suffixes, and circumfixes. However, unlike Arabic, German has many verbs (called "weak verbs") which are regular in the sense that they show no stem change at all. The irregular verbs, or "strong verbs", show many different patterns of stem changes. Another difference to Arabic is that the affixes are not the same for all verb paradigms. In particular, the weak verbs form several inflectional classes (which, of course, differ only in affixes). Finally, unlike Arabic, in the strong verbs the orthographic root does not necessarily consist

GER Inflectional Class (IC) Repository						
	IC-1		IC-2		IC-3	
PI1S		□o□ +e		□e□ +e		□e□ +e
PI2S		□o□ +st		□ie□ +st		□i□ +st
PI3S		□o□ +t		□ie□ +t		□i□ +t
PI1P		□o□ +en		□e□ +en		□e□ +en
PI2P		□o□ +t		□e□ +t		□e□ +t
PI3P		□o□ +en		□e□ +en		□e□ +en
PS1S		□o□ +e		□e□ +e		□e□ +e
PS2S		□o□ +est		□e□ +est		□e□ +est
XI1S		□o□ +te		□a□ +		□a□ +
XI2S		□o□ +est		□a□ +st		□a□ +st
XS1S		□o□ +te		□ä□ +e		□ä□ +e
XS2S		□o□ +test		□ä□ +est		□ä□ +est
PP	ge+	□o□ +t	ge+	□e□ +en		□e□ +en
INF		□o□ +en		□e□ +en		□e□ +en

GER Lexicon					
IC-1		**IC-2**		**IC-3**	
holen	**sohlen**	**sehen**	**lesen**	**vergeben**	**begeben**
‖h,l‖	‖s,hl‖	‖s,h‖	‖l,s‖	‖verg,b‖	‖beg,b‖
'fetch'	'sole'	'see'	'read'	'forgive'	'occur'

Table 4: Example of three inflectional classes (some cells omitted in the interest of space economy) and associated lemmas with their roots in German ("X" stands for past tense). The different blocks in the IC table specify different stem syncretism zones.

of sequences of single letters: the strong verbs have monosyllabic stems (plus perhaps derivational morphology), with the vowel in this stem potentially undergoing changes. However, the onset and coda of the stem syllable can be any consonant cluster allowed by German phonology. Thus, in German, we model roots as pairs of strings of any length (which represent the onset and coda of the stem syllable). Table 4 shows a weak IC and two strong ICs.

Since German strong verbs can have any stem vowel, we assume that all eight vowel letters of German (*aeiouäöü*) are pattern letters, and all other consonant letters are root letters. German weak verbs show no stem changes at all; if we tailored our templates to them, we would define all letters to be root letters, and there would be no pattern letters at all. This is in fact the experiment we reported on in Section 4.6, and whose results are shown as NOTMP in Table 5. However, since we do not know during training time whether a verb is weak or strong, all verbs will be modeled with an orthographic template, even though there is no stem change at all.

Empirically Determined Pattern Letters For GER, the EMPR approach, using the algorithm described in Section 5.2, yields *oaäß* as the optimal

set of pattern letters. The EMPR pattern letters differ from the SCHLR pattern letters by omitting five vowels, but including the *ß* variant of the *s*.

Results In table 5 we see that using all eight vowels as pattern letters (SCHLR column) in fact decreases performance at every training size (and relatively more at smaller training sizes). However, if we use the empirically obtained pattern letter set *oaäß*, we see that we perform better than SCHLR at almost all training sizes, and slightly better than NOTMP at larger training sizes.

Error Analysis A manual inspection of all development error types again revealed 8% development set annotation errors and 4% acceptable variations. To investigate why NOTMP outperforms SCHLR for German on the development set, we performed an oracle experiment: we assumed we knew for each seen verb in training whether it is a weak or a strong verb. If it is weak, we model it using NOTMP, and if it is strong, using SCHLR. We observe as expected that the performance on weak verbs is very similar to that obtained using NOTMP on all verbs. However, for the strong verbs, it is only when we have more than 75,000 words of training data that the oracle outperforms NOTMP. We assume that the reason is that the highly frequent verbs are strong, but occur frequently enough so that their IC can be learned directly from the training data. Using SCHLR simply adds noise for these very frequent verbs. It is only for the less frequent strong verbs that SCHLR can contribute, and then only when a large amount of training data is available.

5.4.2 Blind Test Set

Table 6 shows the accuracies the different systems on our blind test set. The results for the test set are lower than those of the development set, and NOTMP, SCHLR,and EMPR produce very similar accuracy results. We also compare our results to those obtained by running the system of Durrett and DeNero (2013) on the same training and test data. Our system outperforms Durrett and DeNero (2013)'s system reducing the error of by 5%.[5]

[5]We also tested our system on Durrett and DeNero (2013)'s problem definition and data, training on 200 GER paradigms, and testing on 200 unseen paradigms. This is the case of testing for unseen lemmas in our system. Our system gives an accuracy of 88.4% as opposed to 91.8% as reported by Durrett and DeNero (2013). Our system was not designed for this task.

Corpus Size	Verb Count	BL	NOTMP	SCHLR	EMPR
0K	0	13.7	25.2	25.2	25.2
1K	81	43.3	72.0	67.0	69.4
5K	461	64.8	83.5	83.0	83.1
10K	929	72.1	89.0	87.4	88.6
25K	2,362.0	79.5	93.6	93.0	92.4
50K	4,527	86.2	95.6	95.1	95.6
75K	6,728	88.9	96.8	96.2	97.0

Table 5: Learning curve comparing system performance for GER on **tokens** on DEV corpus. BL=Baseline

System	Accuracy	Error Reduction
Baseline	89.5	
NOTMP	96.5	66.7
SCHLR	96.5	66.7
EMPR	96.5	66.7
Durrett	96.3	64.8

Table 6: Results for GER on **tokens** using a blind test set.

6 Conclusion and Future Work

We presented a method for automatically learning inflectional classes and associated lemmas from morphologically annotated corpora. In the future, we plan to improve several aspects of our models, in particular, using more powerful language-independent template transformations to automatically optimize for stem and affix modeling. We plan to take the insights from this paper and apply them to new dialects and languages with limited resources. We are interested in extending our approach to languages with different morphological systems, e.g., agglutinative or reduplicative. We will explore ideas from unsupervised morphology learning to minimize the need for morphological annotations.

Acknowledgment

This paper is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-12-C-0014. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

References

Sabine Brants and Silvia Hansen. 2002. Developments in the tiger annotation scheme and their realization in the corpus. In *In Proceedings of the Third Conference on Language Resources and Evaluation LREC-02. Las Palmas de Gran Canaria*, pages 1643–1649.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Maris Camilleri. 2011. Island morphology: Morphology's interactions in the study of stem patterns. *Linguistica*, 51:65–84. Internal and External Boundaries of Morphology.

Burcu Can and Suresh Manandhar. 2012. Probabilistic hierarchical clustering of morphological paradigms. *EACL 2012*, page 654.

Erwin Chan. 2006. Learning probabilistic paradigms for morphology in a latent class model. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology at HLT-NAACL*, pages 69–78.

Lionel Clément, Benoît Sagot, and Bernard Lang. 2004. Morphology based automatic acquisition of large-coverage lexica. In *LREC 04*, pages 1841–1844.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1).

Silviu Cucerzan and David Yarowsky. 2002. Bootstrapping a multilingual part-of-speech tagger in one person-day. In *Proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7.

Grégoire Détrez and Aarne Ranta. 2012. Smart paradigms and the predictability and complexity of inflectional morphology. *EACL 2012*, page 645.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 616–627.

Greg Durrett and John DeNero. 2013. Supervised Learning of Complete Morphological Paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Ahmed El Kholy and Nizar Habash. 2010. Techniques for Arabic morphological detokenization and orthographic denormalization. In *Proceedings of LREC-2010*, May.

Ramy Eskander, Nizar Habash, Ann Bies, Seth Kulick, and Mohamed Maamouri. 2013. Automatic correction and extension of morphological annotations. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 1–10, Sofia, Bulgaria, August. Association for Computational Linguistics.

Raphael Finkel and Gregory Stump. 2002. Generating Hebrew Verb Morphology by Default Inheritance Hierarchies. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 9–18.

Markus Forsberg, Harald Hammarström, and Aarne Ranta. 2006. Morphological lexicon extraction from raw text data. *Advances in Natural Language Processing*, pages 488–499.

Hassan Gadalla, Hanaa Kilany, Howaida Arram, Ashraf Yacoub, Alaa El-Habashi, Amr Shalaby, Krisjanis Karins, Everett Rowson, Robert MacIntyre, Paul Kingsbury, David Graff, and Cynthia McLemore. 1997. CALLHOME Egyptian Arabic Transcripts. In *Linguistic Data Consortium, Philadelphia*.

Hassan Gadalla. 2000. *Comparative Morphology of Standard and Egyptian Arabic*. LINCOM EUROPA.

Nizar Habash and Owen Rambow. 2006. MAGEAD: A morphological analyzer and generator for the Arabic dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 681–688, Sydney, Australia.

Nizar Habash, Owen Rambow, and George Kiraz. 2005. Morphological Analysis and Generation for Arabic Dialects. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 17–24, Ann Arbor, Michigan.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012. A Morphological Analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pages 1–9, Montréal, Canada.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350.

H. Kilany, H. Gadalla, H. Arram, A. Yacoub, A. El-Habashi, and C. McLemore. 2002. Egyptian Colloquial Arabic Lexicon. LDC catalog number LDC99L22.

Kimmo Koskenniemi. 1983. Two-Level Model for Morphological Analysis. In *Proceedings of the 8th In-*

ternational Joint Conference on Artificial Intelligence, pages 683–685.

Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. 2008. Paramor: Finding paradigms across morphology. *Advances in Multilingual and Multimodal Information Retrieval*, pages 900–907.

Sylvain Neuvel and Sean A Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 31–40. Association for Computational Linguistics.

Matthew G Snover, Gaja E Jarosz, and Michael R Brent. 2002. Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 11–20.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio, June.

Gregory T. Stump. 2001. *Inflectional Morphology. A Theory of Paradigm Structure*. Cambridge Studies in Linguistics. Cambridge University Press.

Géraldine Walther. 2011. Measuring morphological canonicity. *Linguistica*, 51:157–180. Internal and External Boundaries of Morphology.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216.