1. Introduction

Languages use morphological systems to convey syntactic and semantic meaning. An example of this is word inflection, the process of word formation where a word is modified to express different grammatical categories such as tense or definiteness. Often word inflection is expressed with word affixation (e.g. adding a prefix, infix or suffix) although it is not limited to this method. The degree to which morphological systems are used to convey information depends on the language, and (broadly speaking) a group of languages belonging to the same linguistic topological family could be considered to have similar (morphological) preferences. For instance, a language such as English (from the Analytic family) has only a minor dependence on inflections to distinguish relationships, instead relying on the word order and the use of additional word types (e.g. prepositions); and Vietnamese (Analytic Isolating) has no inflectional morphology whatsoever. In contrast, Turkish (from the Agglunative family) relies almost completely on word inflection through ‘gluing together’ affixes to convey grammatical features.

Morphological reinflection is the task of generating a word inflection from a lemma (source word). Given the diversity of morphological systems, it presents many hurdles to computational approaches. However the often systematic relationships between inflected forms (i.e. patterns) also make it a task suited such methods. The SIGMORPHON shared task series is devoted to this challenge. Datasets, in the form of limited inflection tables paired with a universal annotation schema (UniMorph Schema), for a wide variety of languages are made available each year for people to develop and implement computational methods of morphological re-inflection. Using these datasets, the task is to implement a model that can predict a source word (lemma) not present in the dataset.

This report details an attempt at implementing a limited version of the 2017 SIGMORPHON shared task. The aims are to investigate several alignment and finite-state approaches to morphological reinflection, evaluating their strengths and weaknesses. To this end, a simple ‘word alignment’ affixation method and a linguistically inspired ‘abstract paradigm’ method are implemented built from the ground up, are presented (section 3), with their short comings and potential improvements discussed (Section 4).

2. Related Work

There is a relatively large body of existing work on the NLP task of morphological re-inflection. Prior to 2017, many of the proposed systems took a non-neural machine learning approach and had varied success. Examples of the differing approaches used to extract ‘inflectional rules’ include conducting word alignment based affix identification (Durrett and DeNero, 2013), and utilising linguistic heuristics such as abstract paradigms (Sorokin, 2016; and Ahlberg et al, 2014 and 2015). The latter method is discussed in detail in section 3.
Viewing through a wider lens there is much related work dating back many decades, involving edit distance (and stochastic edit distance), dynamic programming, finite state machines and regular expressions. Details of this and related work can be found in “Speech and Language Processing” (Jurafsky and Martin).

In recent years, most of the work on computational morphological reinflection has involved the use of neural networks, and has enjoyed more success than the non-neural approaches. The 2016 SIGMORPHON report presents the performance gap between neural and non-neural clearly, stating “The neural systems were the clear winner in the shared task. In fact, the gains over classical systems were quite outstanding.”

3. Experiments

Two baseline methods were provided for initial investigation into this task. These came from the SIGMORPHON Shared Task challenge in 2016 and 2017.

The 2016 baseline model incorporates a deterministic (perceptron) classifier to map each lemma to an estimated word form given the UniMorph tag set. To do this, the training set pairs symbol are aligned iteratively with the aim of finding a minimum edit distance. From this, a set of actions are 'learned' (where an action may be made up of one or more 'insert', 'delete', 'repeat', 'change'). The classifier is then used to apply a relevant action during test time by moving left to right through the input test lemma.

The 2017 baseline model, also deterministic, takes a slightly different approach. It is based on a submission to the 2016 shared task by Liu and Mao (University of Colorado) and its implementation could be considered simpler (than the 2016 baseline). First, considering the training set, it aligns the source and target words using Levenshtein distance. Once aligned, it forms a grouping of pre / stem / suffix for the pair and uses this to generate a set of actions (one for prefix generation, one for stem + suffix generation) based on some simple heuristics. It does not account for any other morphological changes that may be characteristic of a language.

Baseline System

In the interest of understanding the challenge of morphological reinflection in diverse language groups, I attempted to implement baseline deterministic model that relies upon word alignment, and takes inspiration from the 2016 University of Colorado system (Liu and Mao, 2016) as well as the 2017 baseline model. The program outline as follows:

Train

- For each pair of words in the training set, source and target are aligned by their longest common sub-string. This is found by:
    - generate all substrings of the smaller string
    - iterate through a sorted (by length) list of these substrings until a match is found within the longer word
    - return each word as a list of prefix / match (stem) / suffix
- From this, prefix and suffix changing actions are extracted and stored (separately) in a 'Rule' object dictionary for later application, so that at the end of training there exists n 'rule' objects and dictionaries where n = # unique UniMorph tag sets in the dataset.

Test

- Split the test set into subsets based on the UniMorph tag. These are treated individually
- For each source word in the test set, treat a suffix change and prefix change as a separate task:
    - Suffix change: Look for a matching suffix change stored in the rule dictionary, preferencing longer matches. For a given length, preference suffix change observed more often in the training set.
    - Prefix change: Look for a matching prefix change stored in the rule dictionary, preferencing changes that were observed more often in the training set.
- By default, the algorithm preferences suffix-changes (given the majority of languages preference). If a training set is found to be predominantly prefix changing, all words in the train and test sets are reversed (so that priority is given to prefix changing without requiring modifications to the algorithm).

Results of this baseline model are discussed in section 4.

“Abstract Paradigm” Implementation

Following on from developing a simplistic deterministic baseline system, two approaches were considered: either look to improve this baseline system, or implement a different model that approaches the problem from a different perspective. While the baseline system is a simple system, easy to understand, it is also quite restrictive (see section 4). As such, the latter approach was chosen; a system was created that instead utilised a probabilistic classifier to predict ‘Abstract Paradigms’ representing word inflections. In what follows is a brief overview of the theoretical basis for the abstract paradigm, as well as an outline of the implementation.

Given an inflection table (in our case, a pair of source and target word forms), it is a reasonable interpretation to consider them as having two components: components that are specific to that word in particular, and components that are common to the inflection transformation (the inflection pattern). This approach has been considered since the mid 20th century (Hockett, 1954).
By replacing the word pair with an ordered set of variables (representing the first component) and sub-strings (representing the second), we can obtain an ‘abstract paradigm’ that represents a certain inflection pattern. For example:

$$convolute, deconvolved => x_1 + ute, de + x_1 + ved$$

$$\text{where } x_1 = convol$$

Here the variable is able to abstract away the component of the pair that is dependent on the word itself, so that we are left with the components that are specific to this particular inflection pattern.

Compared to the baseline approach of word-alignment pre and suffix changes only, an abstract paradigm representation can capture language inflection patterns with more flexibility, resulting in (in principle) a superior method to approach the task of morphological inflection.

Program outline

In a nutshell, the program uses a sequence alignment approach to generate and predict inflection patterns, using these in turn to predict the target word form. It draws influence from the SIGMORPHON 2016 MSU submission system (Sorokin, 2016) as well as the research into learning of morphological paradigms (Ahlberg, Forsberg, Hulden 2014 and Ahlberg 2015).

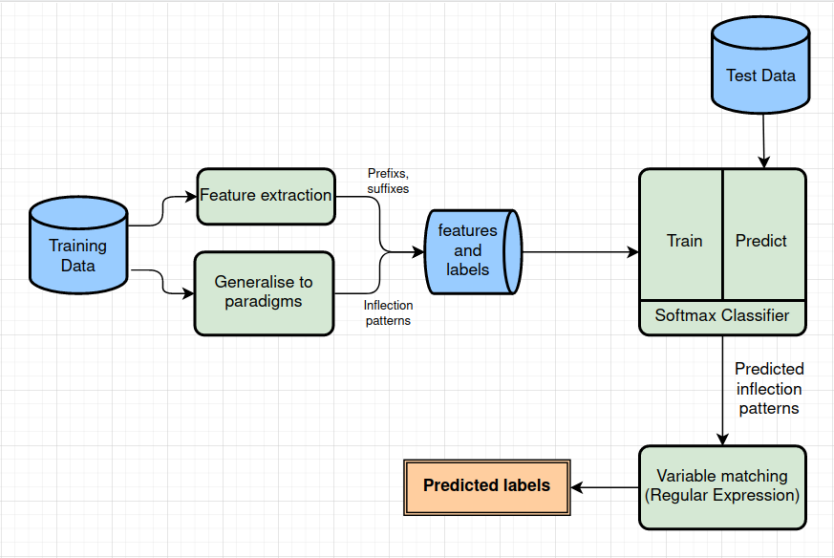

Figure 1: System architecture overview

The system considers each morphological tag set token as an independent problem, and a ‘rule’ object is created for each (each is trained independent of the other). The core components of the system are: generalisation and generation of paradigms, classification training, and paradigm matching. Each of these are discussed below.

Generalisation and generation of paradigms

For each word form pair in the training set for a given token, the longest common subsequence (LCS) is found. The LCS is considered an ordered (but not necessarily consecutive) sequence of characters common to both words.

Once the LCS is found, substrings of the LCS are matched (greedy) on both source and target words with criteria that they must be consecutive. When a match is found, the LCS substring is stored as a variable, and in each of the word forms it is replaced by the variable placeholder

Finally, the abstract representation is generated with variables being integers and the inflection pattern components remaining as substrings.

	source	target	paradigm	rule
0	yan cümle	yan cümlemize	0#0mize	N;DAT;SG;PSS1P
1	sempozyum	sempozyumumuza	0#0umuza	N;DAT;SG;PSS1P
2	fonksiyon	fonksiyonumuza	0#0umuza	N;DAT;SG;PSS1P
3	tütsü	tütsümüze	0#0müze	N;DAT;SG;PSS1P
4	eşek	eşeğimize	0k#0ğimize	N;DAT;SG;PSS1P
5	sululuk	sululuğumuza	0k#0ğumuza	N;DAT;SG;PSS1P

Figure 2 – Abstract Paradigms for a set of tags (Turkish)

Classification training

Each source word in the training set has prefixes and suffixes extracted by simply considering all consecutive character subsets of the first *n* and last *m* letters (following the MSU submission by A. Sorosky, default values are n=3, m = 5). These are added as features (one hot encoding) to the dataset, along with the pairs ‘inflection pattern’ (abstract paradigm) found during the generalisation step above. A softmax classifier (Sklearn implementation) is trained to predict the inflection pattern. Regularisation of the classifier is kept to a minimum. (Specific parameters: solver=lbfgs, regularisation constant=10).

Paradigm Matching

Once the classifier is trained, it is used to predict an inflection pattern (in the form of an abstract paradigm) for a given lemma in the test set. This must then be matched to the lemma in order to obtain values for the variables. A naive regular expression builder function iterates through the paradigm and generates a regular expression, e.g.:

$$0a1ed => (\D+)a(\D+)ed$$

The regular expression is used to match the variables (0 and 1 in this example) to their components. Once this is done it is straight forward to apply the inflection pattern to generate the predicted form.

4. Evaluation and Error Analysis

The languages chosen for this task were Turkish (Turkic, agglunative), Macedonian (Slavic, fusional), and Hebrew (Uralic, templatic). English was used initially in developing the implementations as it was though that debugging would be more interpretable (however in hindsight this was a poor decision, discussed further below).

Development Set

Accuracy

Model	**English**		**Turkish**		**Macedonian**		**Hebrew**	
	Low	Medium	Low	Medium	Low	Medium	Low	Medium
Baseline	63.4%	61.0%	11.9%	28.5%	34.1%	51.2%	24.3%	38.3%
Paradigm	61.9%	61.9%	14.5%	32.4%	42.2%	51.6%	22.3%	31.3%

Average Levenshtein Distance

Model	**English**		**Turkish**		**Macedonian**		**Hebrew**	
	Low	Medium	Low	Medium	Low	Medium	Low	Medium
Baseline	0.447	0.488	3.982	2.998	1.434	0.721	1.176	1.0
Paradigm	0.443	0.443	4.211	2.053	1.245	0.761	1.566	1.131

Test Set

Accuracy

Model	**English**		**Turkish**		**Macedonian**		**Hebrew**	
	Low	Medium	Low	Medium	Low	Medium	Low	Medium
Baseline	63.2%	63.0%	13.2%	25.6%	37.5%	51.9%	20.9%	39.5%
Paradigm	64.7%	64.7%	14.6%	34.2%	43.0%	52.9%	17.6%	32.3%

Average Levenshtein Distance

Model	**English**		**Turkish**		**Macedonian**		**Hebrew**	
	Low	Medium	Low	Medium	Low	Medium	Low	Medium
Baseline	0.43	0.439	5.761	3.041	0.985	0.698	1.451	0.98
Paradigm	0.406	0.406	4.478	2.065	1.174	0.742	1.603	1.076

Overall, the performance of both models was poor considering the average system performance of the submitted systems to the 2016/2017 shared task. Results of note:

- Both systems performed the best on English, which has the least variety in inflections.
- Somewhat disappointingly, there was minimal improvement in the paradigm system over the baseline system (if at all- it actually performed worse for Hebrew).
- There was a decrease in performance as the training set size was increased for English, for both systems. This is particularly odd behaviour (discussed further below).

Despite considerable additional complexity and challenges implementing an abstract paradigm based approach, the system performance was comparable to that of the baseline system.
Investigating this further, it is suspected that the relatively poor performance of the paradigm system is not due to the approach itself, whose additional flexibility and predictive power should be able to make predictions with lower degree of error, but rather a number of implementation issues. Three main issues have been identified, although more may exist.

First, the paradigm generation implementation relies upon finding the longest common subsequence, and then subgrouping this into consecutive characters (forming the variables). There could be several ways to implement this; in our implementation we chose a naive & greedy approach with no constraints on space between the groups or the length of characters in a group. This results in erroneously matching on single characters with a large gap between variables:

	source	target	paradigm	rule
0	fortlamak	fortlamıyorlar mıydı?	01k#0ıyorl1r mıydı?	V;IND;3;PL;PST;PROG;NEG;INTR
1	kötüleşmek	kötüleşmiyorlar mıydı?	0ek#0iyorlar mıydı?	V;IND;3;PL;PST;PROG;NEG;INTR

Figure 3: paradigm generation errors

We see variable ‘1’ is matched on the single ‘a’ in the first word, which is arguably not a correct match. In future, adding constraints to the paradigm matching process to limit this kind of matching will help to generate patterns that have more applicability.

The second issue identified is that of the classification process. This implementation created prefix and suffix features based on every consecutive substring of <3 characters for prefix and <5 characters for suffix (e.g. “history” has {(h, hi, his), (y, ry, ory, tory, story)} and did not do any filtering or aggregating of frequent occurrences. Systems that performed better than ours remarked on the importance of this step (Sorokin, 2016). Without this, the high occurring features skewed the classifier which hindered its flexibility, causing it to often predict a blanket rule:

Lemma	True Inflection	Prediction
saat	saatlerden	saatlardan
robot	robotlardan	robotlardan
yaya	yayalardan	yayalardan

Figure 4: classification errors

In future, we would recommend developing a feature selection process that eliminated frequent prefixes at a minimum, and perhaps investigate a decision tree based approach for ease of interpretation and tuning.

Third, the paradigm matching process relied upon a naive approach of building a regular expression ‘builder’ that matched greedily and had very minimal constraints placed on it. This created errors such as repeated characters when the inflection pattern was applied.

Perhaps the largest single contributor to the low performance of these models, however, is that they were developed testing on the English datasets alone. This approach was taken as it was thought that while implementing and debugging, it would be easier to interrogate if the dataset was in a language that the author was fluent in. However, this short-sighted approach caused a great many use cases to be missed, such as the simple case of test-set tags being unseen in the training set. It also gave false confidence during the initial stages (as ~60% accuracy was thought to be a good start). Many design decisions would have been different had a multi-language approach been adopted from the initial stages.

Conclusion

A word's form reflects syntactic and semantic features that are expressed by the word. Inflections such as affixation are commonly used within diverse languages to convey grammatical distinction, and these inflections tend to follow a systematic pattern. Morphological reinflection is the task of generating a word inflection from a lemma (source word), and is an important and challenging task of Natural Language Processing. Over the past several decades there have been a variety of approaches to tackling morphological reinflection, including finite state transduction and alignment based methods as well as neural network approaches.

The goal of this assignment was to investigate and develop an understanding of alignment and finite state approaches to NLP tasks. To this end, the author developed two naive, 'proof of concept' models from the ground up- both taking a sequence alignment approach yet differing substantially in their modelling and prediction process; evaluating the performance of each on a subset of the SIGMORPHON 2017 shared task (part 1). The first model, a baseline system, is a fully deterministic system using word alignment rule-extraction; while the second utilised a linguistic approach to modelling inflection patterns called 'abstract paradigms' that offered more flexibility, and combined this with a multiclass logistic regression classifier (softmax) to predict inflected forms.

Performance of either system was poor when compared to the non-neural models submitted to the SIGMORPHON 2016 task, and there is little doubt that this is due to their simplistic implementations limited by both time and (the author having only) a developing knowledge of language processing systems. However, the process of designing and implementing the two models brought with it many insights into the challenge of finite-state (and, more broadly, computational) approaches to language, and in this regard they performed very well in meeting the assignment goals.

References

1. *Semi-supervised learning of morphological paradigms and lexicons.* (Malin Ahlberg, Markus Forsberg, and Mans Hulden, 2014)

2. Paradigm classification in supervised learning of morphology. (Malin Ahlberg, Markus Forsberg, and Mans Hulden, 2015)

3. Inflection generation as discriminative string transduction. (Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak, 2015)

4. Supervised learning of complete morphological paradigms. (Greg Durrett and John DeNero, 2013)

5. Using longest common subsequence and character models to predict word forms (Sorokin, 2016)

5. Speech and Language Processing 2ed (Jurafsky, Martin, 2008)

6. *Morpholog-ical reinflection with conditional random fields and unsupervised features.* (Ling Liu and Lingshuang Jack Mao, 2016)