An Exploration of Datalog Applications to Language Documentation and Reclamation

Anita Baral[1], Pratiksha Shrestha[1], Jitendra Sharma[1], Hunter Lockwood[1,2] and Daniela Inclezan[1]

[1]Miami University, Oxford, OH USA
[2]Myaamia Center, Oxford, OH USA

Abstract

Language documentation and reclamation involve the systematic recording, preservation, analysis, and revitalization of endangered or dormant languages to safeguard linguistic diversity and cultural heritage. This paper explores the application of Datalog in these efforts, with a particular focus on the Myaamia language. We introduce a stem decomposition tool designed for the Myaamia language, highlighting its contributions to both the documentation process and the testing of linguistic hypotheses. Our tool uses the DLV system.

Keywords

DLV, linguistics, Language Documentation and Reclamation, Datalog

1. Introduction

This paper presents an initial exploration of Datalog applications to Language Documentation and Reclamation (LDR). LDR involve the systematic recording, preservation, analysis, and revitalization of endangered or dormant languages, necessary to safeguard linguistic diversity and cultural heritage. Our work focuses on the Myaamia language, a reawakening language in the Algonquian family which today is used by a growing number of second language speakers, including members of the Miami Tribe of Oklahoma. The Myaamia Center, a Miami Tribe of Oklahoma initiative located within Miami University, has been developing the Indigenous Languages Digital Archive (ILDA), an online platform where communities are in control of the analysis and display of materials at every step of the process. In our work, we will reference the ILDA dictionary for Myaamia language[1] and the database that supports it. We focus on an application of Datalog to Myaamia stem decomposition, which informally refers to the identification of the smallest constituents carrying lexical meaning that combine to form a word.

The structure of Datalog presents advantages to Language Documentation and Reclamation, as scholars in this field tend to be extensively trained on precisely this sort of symbolic computational systems. Moreover, the transparent nature of Datalog programs, as compared with black-box Machine Learning systems, is preferred by the communities that lead LDR efforts and for which data sovereignty is of utmost importance. We decided to use the system DLV[2] for our use case in order to facilitate the addition of more complex rules in the future and leverage the full expressive power of Answer Set Programming [1]. Logic programming has a long history of being used in linguistics and NLP research [2, 3, 4]. For instance, a work similar to ours explores the use of Prolog for Uzbek morphological parsing [5], while other researchers have applied Inductive Logic Programming to the linguistic task of recognizing monosyllables in a language [6]. Our initial stem decomposition tool demonstrated the following contributions to Myaamia language documentation: proposing new word derivation

Datalog 2.0 2024: 5th International Workshop on the Resurgence of Datalog in Academia and Industry, October 11, 2024, Dallas, Texas, USA

barala@miamioh.edu (A. Baral); shrestp9@miamioh.edu (P. Shrestha); sharmaj2@miamioh.edu (J. Sharma); lockwoht@miamioh.edu (H. Lockwood); inclezd@miamioh.edu (D. Inclezan)
0000-0002-4534-9658 (D. Inclezan)

[1]https://mc.miamioh.edu/ilda-myaamia/dictionary
[2]https://www.dlvsystem.it/dlvsite/

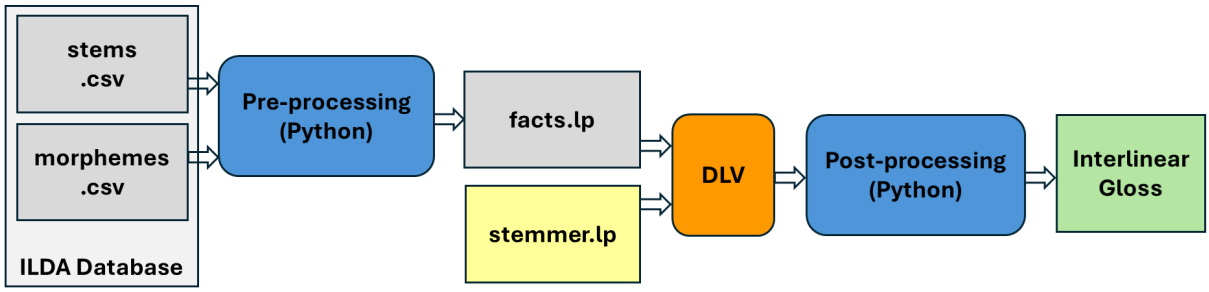

Figure 1: Stem Decomposition Tool for Myaamia Language.

possibilities; enabling testing of linguistic theories; and facilitating the identification of errors in the database for improved documentation.

2. Stem Decomposition for Myaamia Language

Let us start by introducing some linguistic terms. A stem is a part of a word responsible for its lexical meaning. A morpheme is the smallest meaningful constituent of a linguistic expression. An affix is a morpheme that is attached to a word stem to form a new word or word form. Affixes could be derivational (leading to the formation of a new word with a new meaning and possibly a different syntactic category) or inflectional (expressing some grammatical category such as tense, person, number, gender, etc.). Stem decomposition is the process of breaking words into their stem and affixes.

Here, we present a tool for derivational stem decomposition for the Myaamia language, developed using Datalog (specifically the DLV system) and Python. The tool's underlying process is described in Figure 1. Initially, information about stems and morphemes is downloaded from the ILDA Database in the form of csv files (*stems.csv* and *morphemes.csv* respectively) that are pre-processed by a Python script to produce a Datalog program consisting of facts, *facts.lp*. This file together with an additional Datalog program in which we encode the stem decomposition rules of Myaamia language, *stemmer.lp*, are fed together into the DLV system. The resulting answer sets are post-processed by a second Python script that produces a more readable output in a format adapted from linguistics scholars, a so-called *interlinear gloss* [7].

Next, we provide further details about the tool's main Datalog components.

3. Datalog Encoding of Facts about Stems and Morphemes: *facts.lp*

ILDA Database entries for both stems and morphemes consist of a unique ID, the stem/morpheme itself in Myaamia language, its English translation, an identifier denoting its part-of-speech classification, and other fields not relevant to our task. Some examples of stems are *nawi-* meaning *"go get (?)"* and *apikaateesi-* meaning *"warm one's feet."* In Algonquian languages like Myaamia, morphemes can be initial, medial, or final, which is noted in the ILDA Database by a dash symbol at the position where the morpheme may be attached to other morphemes/stems. Some examples of morphemes as specified in the ILDA Database are:

- **initial**: *naw-* meaning *"go and, andative"*; *ap-* meaning *"warm"*
- **medial**: *-ikaatee-* meaning *"foot"*
- **final**: *-i* indicating an *"abstract final"*; *-esi* meaning *"by heat"*

We use the predicate $stem$ to denote Myaamia stems coming from the *stems.csv* file, and the predicates $initial$, $medial$, and $final$ for the three different types of Myaamia morphemes in the *morphemes.csv* file. Each stem or morpheme is represented as a list of characters and the dash symbol is omitted, as shown in the facts below:

```
stem(["n","a","w","i"]).
stem(["a","p","i","k","a","a","t","e","e","s","i"]).
initial(["n","a","w"]).
initial(["a","p"]).
medial(["i","k","a","a","t","e","e"]).
final(["i"]).
final(["e","s","i"]).
```

Additionally, we capture the other pieces of information for stems and morphemes using a predicate *gloss* with four parameters: the Myaamia stem/morpheme, its type (which could be `stem`, `initial`, `medial`, or `final`), its English translation, and its part-of-speech classification. Here are the glosses for the stem and morphemes introduced above:

```
gloss(["n","a","w","i"], stem, "go get (?)", "v.an.intran").
gloss(["a","p","i","k","a","a","t","e","e","s","i"], stem, "warm one's feet",
       "v.an.intran").
gloss(["n","a","w"], initial, "go and, andative", "initial").
gloss(["a","p"], initial, "warm", "initial").
gloss(["i","k","a","a","t","e","e"], medial, "foot", "medial").
gloss(["i"], final, "abstract final", "an.intran.final").
gloss(["e","s","i"], final, "by heat", "an.intran.final").
```

4. Derivational Stem Decomposition: *stemmer.lp*

Our Datalog encoding makes use of the following built-in list predicates from DLV:

- $\#append(X, Y, Z)$ – Z is a list obtained by appending the elements of Y to X
- $\#length(X, Y)$ – Y is the size of list X
- $\#getnth(X, Y, Z)$ – Z is the element at position Y in list X
- $\#delnth(X, Y, Z)$ – Z is a list obtained by deleting the element of list X at position Y

4.1. Primary Derivation

In Myaamia, a stem can be derivationally decomposed according to the following three cases outlined by Goddard [8]:

1. The stem itself is an initial morpheme.
2. The stem is an initial morpheme followed by a final morpheme.
3. The stem is an initial morpheme, followed by a medial morpheme, followed by a final morpheme.

Each case is encoded by a rule for predicates $decompose\_1$, $decompose\_2$, and $decompose\_3$, respectively:

```
decompose_1(S, S)       :- initial(S), stem(S).
decompose_2(S, I, F)    :- #append(I, F, S), initial(I), final(F), stem(S).
decompose_3(S, I, M, F) :- #append(I, M, T), #append(T, F, S), initial(I),
                           medial(M), final(F), stem(S).
```

Next, we define a set of predicates that make the output of the Datalog program more readable. These predicates gather information about the English translation and the part-of-speech classification of the morphemes into which a stem is decomposed, as illustrated by the next rules. Here, variables starting with `T` stand for an English translation and variables starting with `P` denote part-of-speech information:

```
interlinear_gloss_1(S, S, TS, PS, TI, PI) :-
    decompose_1(S, S), gloss(S, stem, TS, PS), gloss(S, initial, TI, PI).
interlinear_gloss_2(S, I, F, TS, PS, TI, PI, TF, PF) :-
    decompose_2(S, I, F), gloss(S, stem, TS, PS),  gloss(I, initial, TI, PI),
    gloss(F, final, TF, PF).
interlinear_gloss_3(S, I, M, F, TS, PS, TI, PI, TM, PM, TF, PF) :-
    decompose_3(S, I, M, F), gloss(S, stem, TS, PS), gloss(I, initial, TI, PI),
    gloss(M, medial, TM, PM), gloss(F, final, TF, PF).
```

The $interlinear\_gloss$ atoms in the DLV output serve as input for the post-processing Python code that generates the actual interlinear gloss in the format illustrated by an example below:

```
nawi = naw                  + i
       go and, andative     + abstract final
       initial              + an.intran.final
       'go get (?)'
```

The first line of the interlinear gloss specifies the decomposition of a stem into morphemes; the second line captures the English translation of the morphemes; the third line represents the part-of-speech classification of the morphemes; the last line is the English translation of the stem to be decomposed.

4.2. Testing Linguistics Theories

The use of Datalog enables linguistics scholars to more easily test theories. We illustrate here a "dropped w" hypothesis, which says that 'w' at the end of a morpheme can be dropped as more morphemes are appended to it in the stem derivation process in Myaamia. We use new predicates $decompose\_2\_w$ and $decompose\_3\_w$ to encode this hypothesis, which allows us to determine whether it leads to any new decompositions. Note that there are three different rules for $decompose\_3\_w$, for the cases when (1) the initial, (2) the medial, or (3) both the initial and medial morphemes end in 'w'.

```
decompose_2_w(S, I, F) :-
    stem(S), initial(I), final(F), #length(I, N),  #getnth(I, N, "w"),
    #delnth(I, N, New_I), #append(New_I, F, S).
decompose_3_w(S, I, M, F) :-
    stem(S), initial(I), medial(M), final(F), #length(I, N), #getnth(I, N, "w"),
    #delnth(I, N, New_I), #append(New_I, M, T),#append(T, F, S).
decompose_3_w(S, I, M, F) :-
    stem(S), initial(I), medial(M), final(F), #append(I, M, T), #length(T, N1),
    #getnth(T, N1, "w"), #delnth(T, N1, New_T), #append(New_T, F, S).
decompose_3_w(S, I, M, F) :-
    stem(S), initial(I), medial(M), final(F), #length(I, N),
    #getnth(I, N, "w"), #delnth(I, N, New_I), #append(New_I, M, T),
   #length(T, N1), #getnth(T, N1, "w"), #delnth(T, N1, New_T), #append(New_T, F, S).
```

These rules led to the identification of 107 additional derivations. Around 45% of these fell somewhere in the range of "plausible (but wrong)" to "absolutely correct" according to Myaamia linguists.

4.3. Secondary Derivation

In Myaamia and other Algonquian languages, secondary stem decomposition occurs when a stem created through primary derivation (as described so far) is further extended with a suffix to form a new stem. There is a limited number of suffixes used for secondary derivation. One such suffix is *-aakan*, which indicates an instrument. We mark these via the predicate $secondary\_derivation\_final$ as in the example:

```
secondary_derivation_final(["a","a","k","a","n"]).
```

The rule for secondary derivation then becomes:

```
sec_decompose(S, S1, F) :-
    stem(S), stem(S1), secondary_derivation_final(F), #append(S1, F, S).
```

An example of a secondary derivation is *niimaakan-* (*"flag"*), decomposed as illustrated by the interlinear gloss below, where *niim-* is an instance of the first case of primary derivation (i.e., the stem itself is an initial morpheme):

```
niimaakan = niim                      + aakan
            Carry him aloft (in hand) + instrument
            v.tran.an                 + n.final
            'Flag'
```

Another example is *maawihšinaakan-* meaning *"band, hunting party,"* obtained through secondary derivation from the stem *maawihšin-* and the suffix *-aakan*:

```
maawihšinaakan = maawihšin                       + aakan
                 head of a band or hunting party + instrument
                 v.an.intran                     + n.final
                 'band, hunting party'
```

where *maawihšin-* is obtained through primary derivation as follows:

```
maawihšin = maaw              + ihšin
            together, collect + be in position, lie
            initial           + an.intran.final
            'head of a band or hunting party'
```

5. Discussion

The stem decomposition tool offers three key contributions to Myaamia language documentation and reclamation:

1. It encourages new perspectives on how words may have been derived.
2. It enables the testing of linguistic theories, such as the "dropped w" hypothesis discussed earlier.
3. It enhances the documentation process by identifying errors or inconsistencies in the ILDA Database.

We include here a few examples of ILDA Database inconsistencies that were detected by the stem decomposition tool.

Example 1: The morpheme *eeli* with the meaning *"by boat"* was marked incorrectly as a medial morpheme in the database via the use of dashes, as in *-eeli-*. Instead, this should have been marked as a final morpheme, stated as *-eeli*. This issue was detected by analyzing the decomposition shown below for the stem *ciileelim-* meaning *"like him, think well of him, think highly of him."* Note the incompatibility between the meanings of the morphemes (second line of the interlinear gloss) and the meaning of the decomposed stem (on the last line of the interlinear gloss):

```
ciileelim = ciil          + eeli          + m
            intense       + by boat       + (abstract final)
            initial       + tran.an.final + tran.an.final
            'Like him, think well of him, think highly of him'
```

The correct decomposition, also identified among the results from our tool, is:

```
ciileelim = ciil    + eelim
            intense + by thought
            initial + tran.an.final
            'Like him, think well of him, think highly of him'
```

Example 2: Some incorrect part-of-speech annotations of morphemes were detected. Such observations can speed up the database cleaning and consolidation process. For instance, the morpheme *alamon-* was discovered to have duplicate entries, one marked as an initial morpheme, the other as an inanimate noun, in terms of the part-of-speech information.

Example 3: Another observation facilitated by our tool was that stems used in secondary derivation were entered into the ILDA Database's morpheme table, along with their part of speech information. This contrasts with the usual practice of labeling morphemes' part of speech information as $initial$, $medial$, or $final$, thus prompting a cleanup of the morpheme table.

6. Challenges and Opportunities

Our initial exploration indicated some areas of improvement for our stem decomposition tool. Some decompositions were deemed incorrect by Myaamia linguists due to a mismatch in parts-of-speech between the morphemes combined to form a stem. For instance, the following decomposition for the stem *nawi-* is incorrect because an 'adverb formative' cannot be added to a verb; it can only be added to an adverb.

```
nawi = naw              + i
       go and, andative + adverb formative
       initial          + final
       'go get (?)'
```

Other decompositions were mistakenly identified as derivational when they were, in fact, inflectional. For example, the tool incorrectly decomposed *alamoni-* (*"red ocher, paint, vermillion"*), as shown below:

```
alamoni = alamon            + i
          ochre, vermillion + abstract final
          n.inan            + an.intran.final
          'red ocher, paint, vermillion'
```

when, in this case, the final *-i* is an inflection for nouns that marks gender.

Some derivations went undetected by our tool because phonological transformations occur when morphemes are attached to stems, adhering to the language's phonological rules. For instance, the decomposition of *apikaateesi-* (*"warm one's feet"*) into *ap-* + *-ikaatee-* + *-esi* was missed by the tool, as the stem appears with only two *e*'s instead of three, following the application of surface phonological rules.

A final source of incorrect decompositions was a meaning mismatch between the individual morphemes assumed to combine to form a stem. Here is one such example (see the second and last lines of the interlinear gloss):

```
niimantam = niim        + antam
            dance       + by mouth
            v.an.intran + tran.inan.final
            'carry it by mouth'
```

The morpheme *niim-* has two potential meanings (i.e., it is a homophone): *"dance"* and *"carry him aloft (in hand)."* The inapplicable meaning was selected in this decomposition. In the future, we intend to leverage the full expressive power of DLV and expand our initial encoding with constraints that would prevent invalid decompositions of this type.

DLV vs DLV2: We encoded stems and morphemes as lists of characters to take advantage of the built-in list predicates present in DLV. However, such lists are difficult to read. We additionally explored the use of DLV2 for the stem decomposition application, as DLV2 provides built-in operations for strings. However, when using strings and DLV2, and working with all stems and morphemes in the ILDA

Database, we experienced decreased performance. The DLV2-based tool would not finish the task even in an hour, whereas our original DLV-based tool completed it in 28 seconds with a maximum memory usage of 1,112 MB on a regular laptop. We acknowledge that this would not be an issue if a single stem would need to be decomposed, which is a valid application, but we note that, for database cleaning or update operations, a more efficient implementation would be desirable.

References

[1] M. Gelfond, V. Lifschitz, Classical Negation in Logic Programs and Disjunctive Databases, New Generation Computing 9 (1991) 365–386. doi:10.1007/BF03037169.

[2] M. Kanazawa, Parsing and Generation as Datalog Queries, in: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, ACL, 2007, pp. 176–183.

[3] D. Inclezan, An application of answer set programming to the field of second language acquisition, Theory Pract. Log. Program. 15 (2015) 1–17. doi:10.1017/S1471068413000653.

[4] P. Schüller, Answer Set Programming in Linguistics, Künstliche Intell. 32 (2018) 151–155. doi:10.1007/S13218-018-0542-Z.

[5] G. Matlatipov, Z. Vetulani, Representation of Uzbek Morphology in Prolog, in: M. Marciniak, A. Mykowiecka (Eds.), Aspects of Natural Language Processing: Essays Dedicated to Leonard Bolc on the Occasion of His 75th Birthday, Springer Berlin Heidelberg, 2009, pp. 83–110. doi:10.1007/978-3-642-04735-0_4.

[6] J. Nerbonne, S. Konstantopoulos, Phonotactics in Inductive Logic Programming, in: M. A. Kłopotek, S. T. Wierzchoń, K. Trojanowski (Eds.), Intelligent Information Processing and Web Mining, Springer, Berlin, Heidelberg, 2004, pp. 493–502. doi:10.1007/978-3-540-39985-8_58.

[7] S. L. Chelliah, M. Burke, M. Heaton, Using Interlinear Gloss Texts to Improve Language Description, Indian linguistics 82 (2021). URL: https://par.nsf.gov/biblio/10383818.

[8] I. Goddard, Primary and Secondary Stem Derivation in Algonquian, International Journal of American Linguistics 56 (1990) 449–483.