Can Alternations Be Learned? A Machine Learning Approach To Romanian Verb Conjugation

Liviu P. Dinu
Faculty of Mathematics and Computer Science
University of Bucharest
ldinu@funinf.cs.unibuc.ro

Emil Ionescu
Faculty of Letters
University of Bucharest
emilionescu@unibuc.ro

Vlad Niculae
Faculty of Mathematics and Computer Science
University of Bucharest
vlad@vene.ro

Octavia-Maria Şulea
Faculty of Foreign Languages and Literatures
University of Bucharest
mary.octavia@gmail.com

Abstract

In this paper we look at the conjugation of the Romanian verb, in particular, at its irregularities, from a machine learning point of view. Our attempt is to predict the presence or absence of any alternation in the stem (apophony), using n-gram representations of the infinitive. We combine formal labelling mechanisms with learning methods in order to build a general conjugational model.

1 Introduction

The problem that we approached in this paper deals with phonological alternations in the stem of the Romanian irregular verbs during their conjugation. What we attempted to investigate, using machine learning techniques, was whether there is reason to believe that a pattern can be identified in the conjugation of the Romanian verb and whether that pattern could be learnt through automatic means such that, given the infinitive form of a verb, its correct conjugation could be produced.

Like other Romance languages, Romanian has traditionally received a Latin-inspired classification of verbs into 4 conjugational classes (or sometimes 5, where the 4th conjugation is divided between verbs with the infinitive ending in i and î, respectively) based on the ending of their infinitival form alone (Costanzo, 2011). However, this infinitive-based classification has often been found inadequate due to the many conjugational patterns that have been found in each class and to its inability to account for the behavior of partially irregular verbs (those whose stem has a smaller number of allomorphs than the completely irregular such as "a fi") during their conjugation. There have been, thus, numerous attempts throughout the history of Romanian Linguistics to give other conjugational classifications.

Lombard (1955) combined the traditional 4 infinitive-based conjugational classes with the information related to the variation in the suffix received by 1st and 4th conjugational class verbs in the indicative first person singular form and came up with 6 classes. Other classifications based on the way Romanian verbs conjugate include (Ciompec et. al., 1985 in Costanzo, 2011), who proposed 10 conjugations, and Felix (1964) who came up with 12 conjugations by looking at the inflection of the verbs and the number of allomorphs of the stem. Moisil (1960) proposed 5 regrouped classes of verbs and introduced the method of letters with variable values, in his computer scientific effort toward a "mechanical grammar". Papastergiou et al. (2007) have developed a classification from a (second) language acquisition point of view, dividing the 1st and 4th traditional classes into 3 and respectively 5 subclasses, each with a different conjugational pattern, and offering rules for alternations in the stem.

Finally, Barbu (2007) offered a highly comprehensive classification of the verb conjugation in Romanian.This classification was based on a corpus of more than 7000 verbs, representing verbs of contemporary Romanian, and distinguished 41 conjugational classes which cover the whole corpus. The corpus has also been used in the present research.

As stated before, our focus has been on capturing rules of variation in the stem for partially irregular verbs like a aştepta (to wait), which becomes eu aştept (I wait) and el aşteaptă (he waits) in the "indicativ prezent" tense. It can be seen that the letter e in the stem changes to ea during conjugation, this rich morphology making the language seem difficult to acquire. Attempts to for-

Proceedings of Recent Advances in Natural Language Processing, pages 539–544,
Hissar, Bulgaria, 12-14 September 2011.

malize rules from a computer scientific point of view date back to Moisil in 1960. Such (incomplete) rules can be formulated as context-sensitive grammars, since the alternations are determined by the (phonologic) context in which certain characters (phonemes) appear. This lead us to the idea of analyzing the Romanian verbs from a machine learning point of view: what can one find out by looking at n-gram representation of the infinitives?

In the following, we give a brief description of the context-sensitive grammar rules that we've developed based on a slight modification to Moisil's concept of letters with variable values, a discussion of the implementation of these rules as a parser, and the machine learning techniques applied to the infinitives of a particular class of verbs from a selected corpus.

2 Capturing Verb Alternations in Context-Sensitive Rules

For the moment we limited the discussion to verbs ending in -ta, for which Dinu and Ionescu (2011) gave two rules that, according to our findings, cover 80% of the alternations that appear.

The first rule is for the variable letter t_0 and can be described as:

$$t_0 = \begin{cases} [\text{ţ}]\text{in the context}\#\_[\text{i}]\# \\ [\text{t}]\text{in the context}\#\_[\text{e}] \vee [\text{a}] \vee [\text{ă}] \vee [\text{î}] \vee [\Phi]\# \end{cases}$$

The second rule is for the variable letter u_0 and amounts to:

$$u_0 = \begin{cases} [\text{oa}]\text{in the context}\#\_[\text{ă}] \vee [\text{e}]\# \\ [\text{o}]\text{in the context}\#\_[\text{i}] \vee [\Phi]\# \\ [\text{u}]\text{in the context}\#\_\text{stressed vowel}]\# \end{cases}$$

For example, the verb "a purta" (to wear) has the lemma pu_0rt_0. The third person singular "el poartă" (he wears) is matched by the first context for u_0 and the second context for t_0. The second person singular "tu porţi" (you wear) is matched by the second context for u_0 and by the first context for t_0.

These rules can be formulated as a context sensitive grammar $G = (V_N, V_T, \Sigma, P)$ in the following way:

$$P = \begin{cases} \Sigma \to \$\alpha T_0\#|\$\alpha U_0\beta T_0\# \\ \Sigma \to \$\alpha T_0\text{i}\#|\$\alpha U_0\beta T_0\text{i}\# \\ \Sigma \to \$\alpha T_0\text{ă}\#|\$\alpha U_0\beta T_0\text{ă}\# \\ \Sigma \to \$\alpha T_0\text{ăm}\#|\$\alpha U_0\beta T_0\text{ăm}\# \\ \Sigma \to \$\alpha T_0\text{aţi}\#|\$\alpha U_0\beta T_0\text{aţi}\# \\ T_0\# \to \#\text{t} \\ T_0\text{i}\# \to \#\text{ţi} \\ T_0\text{ă}\# \to \#\text{tă}|\&\text{tă} \\ T_0\text{ăm}\# \to \#\text{tăm}|!\text{tăm} \\ T_0\text{aţi}\# \to \#\text{taţi}|!\text{taţi} \\ x\# \to \#x, \text{for all } x \in V_T \\ x\& \to \&x, \text{for all } x \in V_T \\ x! \to !x, \text{for all } x \in V_T \\ U_0! \to \#\text{u} \\ U_0\# \to \#\text{o} \\ U_0\& \to \#\text{oa} \\ \$\# \to \lambda \end{cases}$$

V_N is the set of non-terminals, V_T is the set of terminals (the alphabet), Σ is the starting non-terminal and P is the set of production rules. For the Romanian language, $V_T = \{\text{a, ă, â, b, ...}\}$. α and β are arbitrary strings from V_T^*. Note that this is the reunion of the two grammars G_1 and G_2 given in (Dinu and Ionescu, 2011).

The power of this grammar does not lie in its language: some of its derivations are general enough to accept any string over the alphabet ending in the letter t, for example. However, derivations in this grammar represent a generative process that can build present indicative forms of verbs.

The way verbal forms are parsed by this grammar with regard to the arbitrary α and β is very important. Take the verb "a certa" (to scold or to quarrel). At the second person singular form, in the present indicative tense, it becomes "tu certi" (you scold) which is accurately modeled by the T_0 alternation. However its third person singular form "el ceartă" (he scolds) exhibits an alternation in the stem vowel "e" that is not captured by these rules. The form "ceartă" is however generated by the grammar, as:

$$\begin{aligned} \Sigma \to \$\text{cear}T_0\text{ă}\# \to \$\text{cear}\#\text{tă} \to \\ \to \$\#\text{ceartă} \to \text{ceartă}. \end{aligned}$$

Therefore, we cannot say that the grammar models this alternation. How can we tell? Note that if we would assume that these derivations completely explain the alternations, it would mean that the verb has two allomorphs for the stem, ”cert” and ”ceart”, yet we have no alternating ”e” vowel rule to account for that variation. This leads to the natural restriction that for a verb to be considered fully modeled by the grammar we previously described, α and β need to remain the same during the derivation of all its forms. This is equivalent to saying that the variable letters should be the only alternations in the stem of a partially irregular verb.

Such grammars are hard to control methods that cannot directly solve the problem of conjugating a verb starting from its infinitive form. We used a simplification of this system to assign labels to verbs depending on how they are conjugated and what alternations they present.

3 Labeling Method

The correct derivations in the grammar presented in the previous section can be formulated as regular expressions. Furthermore, we can associate a particular regular expression for each one of the six possible forms of a verb in this tense. For example, the regular expressions for the conjugation pattern of the word ”a cânta” (to sing) at the first person singular is ^(.+)t$, while for the second person singular it is ^(.+)ţi$, therefore catching the t-ţ alternation. Note that the dot accepts any letter in the Romanian alphabet. The restriction is that, for each of the six forms, the value of the capturing groups (the characters captured by the bracketed part of the expressions) remains constant. These groups correspond to all parts of the stem that remain unchanged and ensure that, given the infinitive and the regular expressions, one can produce a correct conjugation. When the stem has no alternation, the expression will contain only one such capturing group that represents the whole stem.

We will use the term ”rule” to refer to a set of six regular expressions describing the conjugation of a verb. We started incrementally adding rules to cover more of the verbs in the dataset, and arrived at a total of 14 rules. We dropped the rules that only covered one or two verbs, and eliminated these verbs from the dataset. We ended up with seven rules covering 616 of the 628 verbs ending in -ta (98.1%).

An example of one such rule, covering the verb ”a tresălta”, is:

Person	Regexp	Example
1st singular	^(.+)a(.+)t$	tresalt
2nd singular	^(.+)a(.+)ţi$	tresalţi
3rd singular	^(.+)a(.+)tă$	tresaltă
1st plural	^(.+)ă(.+)tăm$	tresăltăm
2nd plural	^(.+)ă(.+)taţi$	tresăltaţi
3rd plural	^(.+)a(.+)tă$	tresaltă

It can be observed that the forms of the verb are consistently accepted by the regular expressions of the rule, with the two groups in brackets always having the values ”tres” and ”l”. This rule is the 5th in our line of 7 rules. Below are listed all 7 of them:

- the 1st rule accepts verbs like ”a ajuta” (to help), which has an alternation in the stem of the sort t→ţ due to palatalization (determined by the 2nd person singular suffix ”i”)
- the 2nd rule accepts verbs like ”a exista” (to exist), which has an alternation in the stem of the type s→ş, due to palatalization as well
- the 3rd rule accepts verbs like ”a deştepta” (to awake/arouse), whose stem has an alternation of the type a→ea
- the 4th rule accepts verbs like ”a deşerta” (to empty), with a stem alternation of the type e→a
- the 5th rule accepts verbs like ”a tresălta” (to start, to take fright), with a stem alternation of the kind ă→a
- the 6th rule accepts verbs like ”a desfăta” (to delight), with a stem alternation of the type ă→a in the 3rd person, and ă→e in the 2nd person singular
- the 7th rule accepts verbs like ”a decapita” (to decapitate), which conjugates with ”ez” suffixes (-ez, -ezi, -ează, -ăm, -aţi, -ează)

These rules were run against the dataset of conjugated Romanian verbs (only those ending in -ta), and a label was assigned to each of the distinct infinitives found, such that the end result consists of a dataset of 616 infinitives, each labeled from 0 to

6 depending on how the verb is inflected during conjugation.

4 Posing the Learning Problem

4.1 Objectives

The problem that we are aiming to solve is to determine how to conjugate a verb, given its infinitive form. The traditional infinitive-based classification taught in school does not take one all the way. Many variations exist within these 4 classes.

The rules from the previous section allow us to separate the verbs ending in -ta into more specific classes, knowing their inflected forms. By assigning the correct label to an infinitive of a verb not inlcuded in our dataset, one obtains all required information in order to produce a correct conjugation. We will now tackle the problem of fitting a model that is able to predict this labelling.

The context sensitive nature of the alternations leads to the idea of n-gram representations. A text feature extractor can be tuned to convert a list of verbs into a data matrix. The features of this data matrix are the substrings of length up to n that occur in the data, and the values can be taken either as occurence counts or simply as binary indicators of occurence. While occurence counts are useful in, for example, information retrieval, we have found that for such character-level applications, frequencies are less relevant than occurence, and it is not useful to give larger weight to n-grams that appear more often.

4.2 Approach

In order to get from a list of strings to a data format suitable for machine learning algorithms, we put together a feature extractor that returns a sparse matrix.

The feature extractor takes two parameters: the maximum n-gram size and whether to binarize the features. It is based on a character n-gram analyzer that takes a Unicode string as input and outputs a list of n-grams that constitute it. For the input ”cânta”, and for $n = 3$ it would produce the list ”c”, ”â”, ”n”, ”t”, ”a”, ”câ”, ”ân”, ”nt”, ”ta”, ”cân”, ”ânt”, ”nta”. The second component of the feature extractor is the vectorizer. This takes a list of unicode strings as input, runs the analyzer on all of them, then establishes the ”vocabulary” of features as the set of distinct n-grams outputted by the analyzer. Afterwards, the vectorizer transforms every string in the dataset into a vector of the same size as the vocabulary. If we want to count features, the i-th element of the vector vector will contain the number of times the i-th n-gram appears in the word. If we want binary features, the vector will contain ones and zeros, indicating whether the n-gram appears or not in the word.

The average word length in the set of infinitives ending in -ta is 7.48. The larger we choose n, the more features the model will have, and therefore the more complex it will be. Considering both of these aspects, we decided on using the value $n = 3$.

The model is built as a pipeline. The list of verbs first passes through the feature extractor and is then fed into the classifier. For classification we evaluated Naive Bayes and linear support vector machines. When using counted features, we used multinomial Naive Bayes, while in the case of binarized features we used Bernoulli Naive Bayes. The support vector classifier uses the one-versus-all approach. Due to the limited size of the dataset, all scores are estimated using leave-one-out cross-validation.

The value of the regularization parameter C for the SVM is decided by a grid search. This consists in defining grid points for fixed parameter values, then fitting and evaluating a model for each grid point using cross-validation.

The system was put together using the scikits.learn machine learning library for the Python programming language (scikits.learn). It provides text feature extraction tools as described above, a linear support vector machine implementation based on the efficient liblinear library, and an automatic grid search framework for tuning the parameters.

4.3 Results

We first looked at how the Naive Bayes score varies as a function of n, the maximum n-gram length. The results can be seen in figure 1. Considering the fact that the number of features grows exponentially with n, the value of $n = 3$ seems to offer an acceptable model compexity trade-off versus classification score.

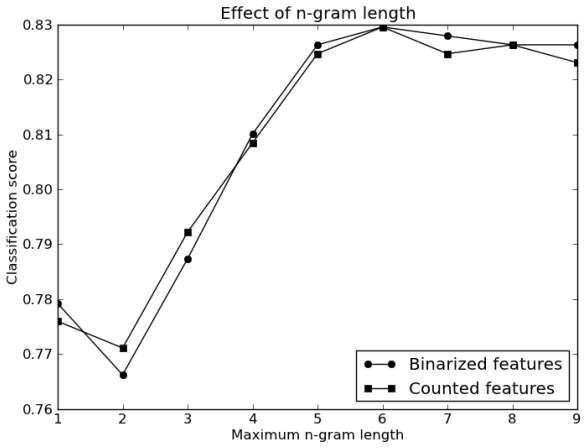

Figure 1: Evolution of Naive Bayes scores as we vary n.

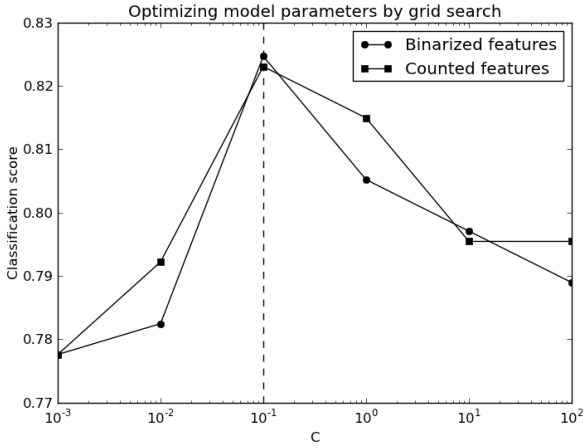

Figure 2: Determining the optimal value for the model parameters

The grid search for optimizing the support vector machine's C, detailed in figure 2 parameter shows that binary features perform better, and the value of C that maximizes the success rate within the grid is found at 10^{-1}, with an accurate classification rate of 82.47%. Precision, recall and F_1 scores for this optimal classifier are presented in table 1. It can be seen that even the poorly represented classes are accounted for. The last class (with label 6), which contains verbs that conjugate without alternations, is the most clearly separated.

class	precision	recall	F_1	support
0	0.65	0.38	0.48	106
1	0.38	0.23	0.29	13
2	1.00	0.60	0.75	5
3	1.00	0.25	0.40	4
4	1.00	0.80	0.89	5
5	1.00	0.25	0.40	4
6	0.85	0.95	0.90	479
avg/total	0.81	0.82	0.80	616

Table 1: Scores estimated by cross validation for the support vector classifier.

The results show that indeed, n-gram based features for classification can give good results for such morphological tasks that are difficult to solve using simple decision rules.

5 Conclusions and Future Works

Our results show that the labelling system based on the verb conjugation model we developed for verbs ending in -ta can be learned with reasonable accuracy. We are currently working on a more extensive system for labelling the infinitives, based on a near exhaustive conjugational model for the Romanian verb.

Our future work will revolve around a more exhaustive classification of the verbs such that, for each class, there is a simple and deterministic way to produce the correct present tense forms, given the infinitive. Following recent works in Romanian linguistics and the study of Romanian as a foreign language such as (Papastergiou et al., 2007), wherein a newer, more comprehensive and in-depth infinitive-based classification of the Romanian verb is given, we aim to extend these results to all verbs, not just the ones ending in -ta, and obtain a usable present tense indicative conjugator for the Romanian language.

6 Acknowledgements

We would like to thank the anonymous reviewers for their insight and constructive criticism which have helped us greatly in polishing this article. We would also like to thank Ana-Maria Barbu for the very useful corpus without which this article wouldn't have existed.

References

Ana-Maria Barbu. *Conjugarea verbelor româneşti. Dicţionar: 7500 de verbe româneşti grupate pe clase de conjugare*. Bucharest: Coresi, 2007. 4th edition, revised. (In Romanian.) (263 pp.).

Angelo Roth Costanzo. *Romance Conjugational Classes: Learning from the Peripheries*. PhD thesis, Ohio State University, 2011.

Liviu Dinu and Emil Ionescu. A context sensitive approach to the problem of phonetic alternations. submitted, 2011.

Jiři Felix. *Classification des verbes roumains*, volume VII. Philosophica Pragensia, 1964.

Alf Lombard. *Le verbe roumain. Etude morphologique*, volume 1. Lund, C. W. K. Gleerup, 1955.

Grigore C. Moisil. Probleme puse de traducerea automată. conjugarea verbelor în limba română. *Studii si cercetări lingvistice*, XI(1):7–29, 1960.

I. Papastergiou, N. Papastergiou, and L. Mandeki. Verbul românesc - reguli pentru înlesnirea însuşirii indicativului prezent. In *Romanian National Symposium "Directions in Romanian Philological Research", 7th Edition*, May 2007.

scikits.learn. scikits.learn, Apr 2011. URL `http://scikit-learn.sourceforge.net`.