Sequence Tagging for Verb Conjugation in Romanian

Liviu P. Dinu and **Octavia-Maria Şulea**
Faculty of Mathematics and Computer Science
Center for Computational Linguistics
University of Bucharest
ldinu@fmi.unibuc.ro
mary.octavia@gmail.com

Vlad Niculae
University of Wolverhampton
vlad@vene.ro

Abstract

Verbs in Romanian sometimes manifest local irregularities in the form of alternating letters. We present a sequence tagging based method for learning stem alternations and ending sequences. Supervised training is based on a morphological dictionary, with a few regular expression paradigms encoded by hand. Our best model improves upon previous machine learning approaches to Romanian verb conjugation, and can generalize to unseen paradigms that can be constructed as variations of the ones in the training set.

1 Introduction

Romanian has a rich inflectional morphology which, in the verbal domain, manifests through complex conjugational patterns. In Table 1, we give an example comparing from left to right: a regular verb, which exhibits an invariable stem, another regular verb, which also exhibits an invariable stem but receives an additional infix *-ez*, a partially irregular verb, which exhibits stem alternation, and a completely irregular verb, which exhibits stem suppletion. The example also shows different syncretism patterns between different conjugated forms. Namely, the 1st and 4th verbs (*a merge* and *a fi*) exhibit 1sg and 3pl syncretism, the 2nd and 3rd verbs (*a dansa* and *a purta*) exhibit 3sg and 3pl syncretism.

Given the richness in ending sequences, stem alternations, and syncretisms, many attempts have been made throughout Romanian linguistics to give conjugational classifications with stronger predictive power than the traditional, Latin-inspired one introduced by Tiktin (1905) which divided verbs into four conjugation classes based on the theme vowel surfacing as the ending in the infinitive form (Costanzo, 2011) and attributed to each of these classes only one general conjugational ending sequence.

The traditional analysis was followed by structuralist ones: Lombard (1955) arrived at 6 classes investigating 667 verbs, Felix (1964) proposed 12 classes, Guţu-Romalo (1968) investigated over 400 verbs and proposed 38 *ending sequences*, which she reduced to 10 verb classes by employing specifically designed homonymy argued against, however, by Avram (1969). When attempting to combine the information gathered about stress shift, ending sequences, and stem alternations, Guţu-Romalo unfortunately ended up with a very extensive classification mirroring a near-exhaustive enumeration of the verbs employed.

More recently, Barbu (2007) distinguished 41 conjugational classes for all tenses and 30 for the indicative present, covering 7,295 contemporary Romanian verbs. Her classes did not take into account stem alternations but only ending sequences, making her classification similar to Guţu-Romalo's 38 ending sequences. On the opposite end, new studies like (Feldstein, 2004) and (Şulea, 2012) take a unifying approach to Romanian conjugation that is elegant in theory but, like

a merge	a dansa	a purta	a fi
to walk	*to dance*	*to wear*	*to be*
merg-λ	dans-ez-λ	port-λ	sunt-λ
merg-i	dans-ez-i	porț-i	eșt-i
merg-e	dans-eaz-ă	poart-ă	est-e
merg-em	dans-ăm	purt-ăm	sunt-em
merg-eți	dans-ați	purt-ați	sunt-eți
merg-λ	dans-eaz-ă	poart-ă	sunt-λ

Table 1: Indicative present conjugation of some Romanian verbs. The first is regular without *-ez*, the next is regular with *-ez*, the next is partially irregular, and the last is fully irregular. We denote the null suffix with λ.

Proceedings of Recent Advances in Natural Language Processing, pages 215–220,
Hissar, Bulgaria, 7-13 September 2013.

many previous approaches, does not lend itself very useful to computational applications.

2 Related work

The first to attempt a computational approach to Romanian morphology was Moisil (1960) who proposed five regrouped classes of verbs, with numerous subgroups. To model stem alternation, he introduced the concept of variable letters, which were letters that changed their value for different forms of the same verb. Following Moisil, Dinu et al. (2011) first implemented a context-free grammar based on alternation rules, using the idea of variable letters. Ultimately, an implementation based on regular expression was used to label the infinitives from a dataset of Romanian verbs conjugated in the indicative present. This was fed into a classifier that attains 90.64% accuracy rate and 89.89% paradigm F_1 score. (Dinu et al., 2012), but in section 3, we point out significant improvements that can be made to this method.

A dictionary-based morphological generator for Romanian was developed by Irimia (2009), based on paradigmatic theory that aims to model roots and suffixes. Access to the resource is restricted. In this paper we attempt a more flexible modelling that covers, in the same way, suffixes and generic variation within the root.

Goldsmith and O'Brien (2006) use neural networks and word-level encodings similar to (Dinu et al., 2011) for learning inflectional classes, but only on highly regular, predictable patterns, with the goal of learning hidden representations, meaningful for psycholinguistic arguments of language acquisition.

Sequence tagging has been successfully used for other morphological applications in recent years. Closest to our application is the application of mined morphological paradigms in (Durrett and DeNero, 2013), the morphological unit segmentation in (Chang and Chang, 2012) and the Finnish morphological generation for machine translation in (Clifton and Sarkar, 2010). A long standing application of such models is the analysis of unsegmented languages, particularly east Asian languages such as Thai (Kruengkrai et al., 2006), Chinese, and Japanese (Nakagawa, 2004).

3 Paradigm overlap and variable letters

In previous work (Dinu et al., 2011; Dinu et al., 2012), we proposed a labelling system that was learned by a linear SVM with 90.64% leave-one-out accuracy. However, when taking a closer look at the labelling rules described, a considerable amount of overlap can be spotted, in terms of what alternations the rules model. Namely, we saw that some rules ended up corresponding to the same variable letter which, however, varied in a different pattern relative to the person and number verb forms. Table 2 illustrates this situation.

rule 10	rule 12	rule 13
a cânta	a deștepta	a deșerta
to sing	*to rise*	*to empty*
^(. *)t$	^(. *)e(. *)t$	^(. *)e(. *)t$
^(. *)ți$	^(. *)e(. *)ți$	^(. *)e(. *)ți$
^(. *)tă$	^(. *)ea(. *)tă$	^(. *)a(. *)tă$
^(. *)tăm$	^(. *)e(. *)tăm$	^(. *)e(. *)tăm$
^(. *)tați$	^(. *)e(. *)tați$	^(. *)e(. *)tați$
^(. *)tă$	^(. *)ea(. *)tă$	^(. *)a(. *)tă$

Table 2: Example of rule overlap in the unstructured system (Dinu et al., 2012)

We noticed that we can treat each word-level paradigm as a set of local variation patterns. These patterns are equivalent to the variable letters introduced by Moisil (1960). Through this reorganisation, several problems with the system from (Dinu et al., 2012) can be alleviated:

- **Class sparsity:** Certain cooccurrences of variable letters are very rare in the dataset, but the individual variable letters may appear more frequently. The global class corresponding to the joint paradigm is difficult to learn due to lack of data. An example is that of the verb *a putea* (to be able to), whose stem vowel *u* transforms into *o* and *oa*, forming a singleton alternation pattern. However, the specific alternation *o-oa* appears in other patterns (*dormi-doarme*).
- **Class interaction:** Word-level classes that include the same variable letters see each other's instances as negative cases and cannot therefore benefit from what they share. By learning each variable letter separately, all occurrences are used as positive cases.

4 Approach

4.1 Available data

Our labelled data is generated from *RoMorphoDict*, an electronic morphological dictionary for Ro-

	T_1	T_2	T_5	T_6	T_{10}	T_{11}	T_{12}	T_{13}
1sg	$	u$	ez$	ez$	$	i$	esc$	iesc$
2sg	i$	i$	ezi$	ezi$	i$	i$	ești$	ierști$
3sg	ă$	ă$	ează	ază$	e$	ie$	ește$	iește$
1pl	ăm$	ăm$	ăm$	em$	im$	im$	im$	im$
2pl	ați$	ați$	ați$	ați$	iți$	iți$	iți$	iți$
3pl	ă$	ă$	ează$	ază$	$	ie$	esc$	iesc$

Table 3: A few of the main ending patterns

manian. The resource is divided according to parts of speech. The subset describing verbs has the following structure for each verb form:

- form
- infinitive
- morphosyntactic description

In (Dinu et al., 2012), we grouped verb forms by their infinitive. We identified, for each of them, six distinct forms covering the two numbers and three persons that are typical of most verbs in Romanian. We wrote sets of six regular expressions that matched paradigms including alternations in the root and could therefore unambiguously describe the conjugation. This is the only place where the morphosyntactic description is used. The matching rules were used as target classes in a one-vs-all multiclass SVM classifier whose input was a bag of all the n-grams within the infinitive, effectively learning to predict the full conjugation paradigm of a verb given its infinitive.

As a follow-up, we propose a finer-grained labelling based on the literature on Romanian conjugation discussed in Section 1. We divided the word-level patterns from in (Dinu et al., 2012) into character-level ones: 16 ending patterns and 17 alternating letters. We used the same regular expressions to identify the verbs that exhibit each combination of patterns and generate labelled instances.

4.2 Sequence tagging

In order to account for multiple interacting variable letters within each verb, we pose verb conjugation as a sequence tagging problem. Each letter in the infinitive is tagged with the particular alternation pattern the verb exhibits for that infinitive letter, or with 0 if the verb exhibits no alternation in that letter during conjugation. Thus, the verb *a tresălta* (to quiver) is labelled as follows:

t	r	e	s	ă	l	t	a
0	0	0	0	a_1	0	t_0	T_1

Here, T_1 encodes the ending pattern received by the class of verbs to which *a tresălta* belongs, as presented in Table 3 along with a few other ending patterns.

4.3 Models and software

The probabilistic model we applied to the verb conjugation problem is a linear-chain conditional random field (CRF). Such models have been often used in NLP because of the linear nature of text: part-of-speech tagging and chunking are important examples of problems that can be successfully solved by sequential prediction models. In the current case, the prediction occurs at the character level, offering a significant computational advantage. The length of a word in letters is usually less than the length of a sentence in words, and the space of possible feature values is also considerably restricted.

Our feature mapping consists of character n-grams to each side of the current letter, up to a fixed window size n, as well as the current letter. The current letter does not form n-grams with the letters around it. For example, the instance of the letter u in *triumfa*, with $n = 2$, would be encoded as:

```
  c[-2]=r c[-1]=i c[-2-1]=ri
c[0]=u c[1]=m c[2]=f c[12]=mf
```

The feature names could just as well be arbitrary, as long as they stay consistent over instances.

The usual way of training CRFs is the maximum likelihood (ML) method (Lafferty et al., 2001). Implementations typically maximize the regularized conditional log likelihood of the data.

Recently, online discriminative methods have been shown to be effective for non-probabilistic training of CRF parameters.

						Cross-validation accuracy			Test accuracy		
method	*ps*	*pt*	n	Θ	N	word	char	char′	word	char	char′
SVM				—		0.886	—	—	0.896	—	—
ML	1	1	4	$\alpha = 0.1$	—	0.924	0.987	0.913	0.914	0.985	0.900
AP	0	1	4	—	10	0.923	0.987	0.917	0.912	0.985	0.900
PA	1	0	4	$C = 1$	10	0.925	0.987	0.917	0.912	0.984	0.900
AROW	1	1	4	$r = 100$	100	0.916	0.986	0.912	0.908	0.984	0.895

Table 4: Results obtained by the best hyperparameter set for each training method. 'word' and 'char' are word-level and character-level scores, respectively. The 'char′' column is the character-level accuracy excluding the '0' class.

The structured averaged perceptron (Collins, 2002) is a simple, fast and effective iterative algorithm. It comes from the even simpler structured perceptron learning algorithm, where at each iteration, a data point (x_i, y_i) is chosen and the model prediction $\hat{y_i}$ is computed. If the prediction is wrong, the model parameters are updated in the direction of the current feature vector.

The averaged perceptron approach takes, instead of the final value of the parameter vector θ, its average $\bar{\theta}$ over all the iterations.

The passive aggressive (PA) algorithm (Crammer et al., 2006) is similar to the averaged perceptron: instead of updating when classification is incorrect, it updates when the margin of the misclassification is more than 1, i.e. when the multiclass structured hinge loss ℓ_t is positive. The update is aggressive in the sense that it forces the new parameter vector to correctly classify the input point with margin of at least 1. Finally, averaging is applied in the same fashion.

The AROW algorithm (Mejer and Crammer, 2010) maintains normal distributions over the parameters of the model and updates their parameters in a way that generalizes PA.

We used *CRFsuite* v0.12 (Okazaki, 2007) for implementation of the learning methods listed above. *CRFsuite* can expand the feature expansion implemented by us at character-level to a vector that optionally includes all possible states (*ps*), all possible transitions (*pt*), or both. These flags, along with the window length n that we have searched for in $\{2, 3, 4, 5, 6\}$, control the feature expansion $f(x, y)$. Apart from this, each algorithm has its own hyperparameters. For ML, we used limited-memory BFGS training with ℓ_2 regularization controlled by α. For AP, we varied the number of iterations N. For PA, we varied N and the aggressiveness parameter C. For AROW, we varied N and the trade-off parameter r. We searched for α, C, r (denoted generally as Θ) over $\{0.01, 0.1, 1, 10, 100\}$ and for N over $\{1, 5, 10, 25, 100\}$. The notations given in parantheses in this paragraph correspond to columns of Table 4.

For more appropriate comparison, we reproduced the word-level SVM results from our previous work (Dinu et al., 2011) but with a held-out test set of a quarter of the labelled data. The best parameters chosen for the linear SVM by 3-fold cross validation on the training set are $n = 8$, $C = 0.15$, *tf-idf* normalization, squared hinge loss and ℓ_2 regularization. The labelling used was the same as in the previous work, with the very small classes discarded, making the problem slightly simpler for the SVM.

5 Results

5.1 Automatic evaluation

We optimized the system hyperparameters using grid search over the parameter spaces described above. The collection of $7,295$ infinitive forms was split into a training set of size $4,699$, a held-out test set of size $2,257$[1], and 339 instances that are still left unlabelled by the identified paradigms.

The validation scores are computed using ten-fold cross-validation over the training set, and the best hyperparameters, in terms of word-level accuracy, for each learning method, are presented in Table 4.

5.2 Manual evaluation

While the previous method verifies that a sequence model benefits from the extra informa-

[1]The split is ad hoc: the first occurrence of any label gets put into the test set, and subsequent occurrences are put into the test set with probability $1/3$. By making sure that all labels are represented in the test set we avoid underestimating the test error.

tion and more accurately reconstructs the conjugation classes for which Dinu et al. (2011) proposed regular expressions, we anticipate that because of higher granularity, a sequence model can give useful results on verbs whose conjugation does not match the predefined patterns. Out of the total of 339 verbs that did not fit into the variable letter and termination patterns that we enumerated, we manually checked the tags given by PA to the first 105 verbs against their actual conjugations (as given in RoMorphoDict). Out of these, 30 had at least one non-null tag correct, demonstrating our method's ability to generalize. The overall tag predictions fell into these categories:

1. completely wrong: neither ending nor alternations (if any) were correctly tagged

2. correct ending, wrong alternations

3. correct alternations, wrong ending.

In terms of wrong endings, the most common mistakes were those when T_1, which represents the tag for the regular conjugational pattern of verbs ending in *-a*, was confused with T_5, the tag corresponding to the standard conjugational pattern of the special class of verbs ending in *-a* which also receive the infix *-ez*. It is likely that the features correlated with these tags are similar, and the tagger thus finds it difficult to choose between the two. We see the same confusion between T_2 and T_6, which are both variations of T_1 and T_5, respectively. And, for the case of verbs with infinitives ending in *-i*, the second largest traditional conjugational class after the first and one which has the *-esc* infix subclass, we see the same type of confusion between T_{10}, T_{12}, T_{11}, and T_{13}. The reason is the same: new verbs, when entering the language, are assigned to either the *-ez* subclass (corresponding to ending tags T_5, T_6) or to the *-esc* subclass (T_{12}, T_{13}) so these classes are the largest in our dataset and, since etymological information is not available, the system cannot tell the difference between these classes.

In terms of alternations, there were 3 verbs which received a correct alternation tag: two which received t_0 and one which received d_0. Both alternations refer to the shift in the 2nd person singular of the letter *t*, respectively *d*, into *ț*, respectively *z*, due to palatalization.

6 Conclusions and future work

We have found that sequential modelling with variable letters is effective for verb conjugation in Romanian. Our system, evaluated on a held-out test set, attains better scores than the leave-one-out results from (Dinu et al., 2011), and furthermore offers greater potential for extensibility to other tenses and modes, through reuse of character-level variations.

After comparing multiple discriminative training methods for CRFs, we have not observed significant variation between their results in terms of accuracy. This is not unexpected, given the small size of the dataset. However, online algorithms lead to much sparser weight vectors: the PA model is almost 40 times smaller than the ML one, and the others are even smaller. Sparse solutions are desired for better interpretability, faster tagging and less overfitting.

A multi-target CRF implementation would permit even more granularity in terms of letter variation, and therefore would be able to learn shared patterns within the same paradigm (i.e. how the variable letter's behaviour in the first person singular influences its behaviour in the first person plural) as well as across tenses and modes. Such models are not readily available in structured learning libraries at the moment since inference in them is costly. For this task, because of the way word lengths are distributed, we expect the problem to be tractable.

Acknowledgements

The contribution of the authors to this paper is equal. Research supported by a grant of the Romanian National Authority for Scientific Research, CNCS – UEFISCDI, project number PN-II-ID-PCE-2011-3-0959.

References

Andrei Avram. 1969. Pe marginea unei morfologii structurale a limbii române ii. *Studii și cercetări lingvistice*, XX(5):557–577. (In Romanian).

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

Joseph Z. Chang and Jason S. Chang. 2012. Word root finder: a morphological segmentor based on CRF.

In Martin Kay and Christian Boitet, editors, *COLING (Demos)*, pages 51–58. Indian Institute of Technology Bombay.

Ann Clifton and Anoop Sarkar. 2010. Morphology generation for statistical machine translation. In *The Pacific Northwest Regional NLP Workshop (NWNLP)*.

Michael Collins. 2002. Discriminative training methods for Hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Liviu P. Dinu, Emil Ionescu, Vlad Niculae, and Octavia-Maria Şulea. 2011. Can alternations be learned? A machine learning approach to Romanian verb conjugation. In *Recent Advances in Natural Language Processing*, pages 539–544.

Liviu P. Dinu, Vlad Niculae, and Octavia-Maria Şulea. 2012. Learning how to conjugate the Romanian verb. Rules for regular and partially irregular verbs. In *European Chapter of the Association for Computational Linguistics 2012*, pages 524–528, April.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the North American Association for Computational Linguistics*, NAACL '13, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ronald F. Feldstein. 2004. On the structure of syncretism in Romanian conjugation. In J. Auger, J. C. Clements, and B. Vance, editors, *Selected Papers from the 33rd linguistic symposium on Romance Languages*, pages 177–195. John Benjamins.

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

John Goldsmith and Jeremy O'Brien. 2006. Learning inflectional classes. *Language Learning and Development*, 2(4):219–250.

Valeria Guţu-Romalo. 1968. *Morfologie Structurală a limbii române*. Editura Academiei Republicii Socialiste România. In Romanian.

Elena Irimia. 2009. Rog – a paradigmatic morphological generator for Romanian. In Zygmunt Vetulani and Hans Uszkoreit, editors, *Human Language Technology. Challenges of the Information Society*, volume 5603 of *Lecture Notes in Computer Science*, pages 74–84. Springer Berlin Heidelberg.

Canasai Kruengkrai, Virach Sornlertlamvanich, and Hitoshi Isahara. 2006. A conditional random field framework for Thai morphological analysis. In *Proceedings of LREC*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Avihai Mejer and Koby Crammer. 2010. Confidence in structured-prediction using confidence-weighted models. In *EMNLP*, pages 971–981. ACL.

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

Tetsuji Nakagawa. 2004. Chinese and Japanese word segmentation using word-level and character-level information. In *Proceedings of the 20th international conference on Computational Linguistics*, page 466. Association for Computational Linguistics.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs).

H. Tiktin. 1905. *Rumänisches Elementarbuch*. Heidelberg: C. Winter. In German.

Octavia-Maria Şulea. 2012. Alternations in the Romanian verb paradigm. Analyzing the indicative present. Master's thesis, Faculty of Foreing Languages and Literatures, University of Bucharest. Available at http://ling.auf.net/lingbuzz/001562.