A Unified Lexical Processing Framework Based on the Margin Infused Relaxed Algorithm. A Case Study on the Romanian Language

Tiberiu Boroș
Research Institute for Artificial Intelligence, "Mihai Drăgănescu", Romanian Academy
tibi@racai.ro

Abstract

General natural language processing and text-to-speech applications require certain (lexical level) processing steps in order to solve some frequent tasks such as lemmatization, syllabification, lexical stress prediction and phonetic transcription. These steps usually require knowledge of the word's lexical composition (derivative morphology, inflectional affixes, etc.). For known words all applications use lexicons, but there are always out-of-vocabulary (OOV) words that impede the performance of NLP and speech synthesis applications. In such cases, either rule based or data-driven techniques are used to automatically process these OOV words and generate the desired results. In this paper we describe how the above mentioned tasks can be achieved using a Perceptron with the Margin Infused Relaxed Algorithm (MIRA) and sequence labeling.

1 Introduction

Natural Language Processing (NLP) applications and Text-to-Speech (TTS) synthesis systems require a set of pre-processing steps that include tasks such as lemmatization, syllabification, lexical stress prediction and phonetic transcription. Because these all these tasks require knowledge of the word composition (derivative morphology, inflectional affixes, part of speech, etc.) we will refer to them as lexical processing steps.

This paper presents a *unified lexical processing framework* based on the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003) designed to solve the basic text-preprocessing tasks involved in both text-to-speech (TTS) synthesis and general NLP applications. Assuming that all existing systems use lexicons for known words, we focused our research in handling the difficult problems generated by presence of out-of vocabulary (OOV) or previously unseen in the training data words that negatively impact the performance of the above mentioned tasks. Our current research is focused on the Romanian language, but the methods presented here are data-driven and with proper lexicons and feature templates, *they can be used for other (Latin based) languages as well*. We show how we achieved state-of-the-art results on Romanian by using the MIRA framework.

2 Lexical processing with MIRA

There are various methods proposed in the literature for each of the previously mentioned lexical subtasks. For each of them, we will offer a short literature review of available methods and we will compare our results with the current state-of-art systems.

The previously proposed methods vary from rule-based to data-driven and different authors employ different classifiers (in data-driven approaches), such as Maximum Entropy Classifiers, Classification and Regression Trees, Support Vector Machines (SVM), Structured SVMs, Conditional Random Fields, etc. While these are all powerful methodologies, we chose the *Perceptron classifier with the MIRA update learning* as our sequence labeling classifier because of its robustness and its ability to obtain highly accurate results that compare to the ones obtained using CRFs. All the lexical processing methods that we propose, share the following similarities:

- All of them are reformulated as sequence labeling tasks;
- We use the same classifier for all our tasks (MIRA);

- The classification context is based on different and mostly lexical (except for lemmatization and lexical stress prediction, which use the morpho-syntactic) feature sets;
- The performance is measured in terms of word accuracy rates (WAR);
- All the tests are reported on OOV words, as we assume that all systems use lookup lexicons for known words;
- All our tests are performed on Romanian and we report the feature sets that yielded the best results.

3 Syllabification

Syllabification is the process of decomposing words into their phonological units, which is an important requirement in modern approaches to TTS synthesis and speech recognition.

All languages have phonetic rules that govern the syllabification process, but it is often the case that these rules are contradicted by etymological principles, a fact which complicates the task of automatic syllabification. Phonetic transcription (letter to sound – L2S) or the position of the lexical stress both provide useful information for syllabification, but more often than not, L2S and lexical stress are not accurate enough on OOV words to help the syllabification process. Also, syllabification lexicons are usually larger than L2S lexicons, thus providing more training data, which helps the syllabification system obtain better results than L2S. Because of the above mentioned reasons, we strictly based our method on purely lexical features (i.e. the word's letters).

Several algorithms have been proposed for the syllabification task divided between rule-based and data-driven. While, rule-based methods are centered on theoretical aspects of the syllabification problem, data-driven methods are usually preferable, since they are language independent and they only require the construction of syllabified words lexicons.

In the following description, we use the term *juncture point* to denote the places where hyphen marks (syllable breaks) are placed within a word.

The look-up procedure was introduced by Weijters (1991). It constructs a table of n-grams from the training corpus and uses this table to predict juncture points. Each n-gram contains the *focus character* (the character that is being analyzed to determine if a juncture point should or should not occur after) with left and right context, including hyphen marks. When syllabification is performed on a new word, the algorithm determines if a focus character should be followed by a hyphen, using the majority of similar n-grams.

The IB1 (Daelemans et al., 1997) algorithm creates n-grams (of predetermined size) from word juncture points and stores them into a database. When a new word has to be split into syllables, every n-gram around the word's possible junctures is matched against the n-grams already available from the training step. N-grams are compared using a distance measure to determine how similar two n-grams are to one another.

Marchand and Damper (2007) introduced Syllabification by Analogy (SbA) which follows the principles of the Pronunciation by Analogy (PbA) algorithm. It works by applying a "full pattern match" on the input string using entries in a dictionary compiled from the training corpora. Marchand and Damper also investigate the possibility of using syllabification to improve grapheme to phoneme performance on English words.

Barlett et al. (2008) use structured SVMs to predict tags for letters in a given word and compare results obtained using different tagging strategies. Their method outperforms the results of the SbA method.

3.1 Syllabification with MIRA

Our sequence labeling approach is inspired after Barlett et al. (2008). In their paper they experimented with different tagging strategies and according to their results, the numbered ONC (onset-nucleus-coda) achieved the highest performance. This is why we employed the same tagging strategy for our system. *The main difference between our approach and theirs, is the features set we designed and the classifier we used (MIRA).*

A widely accepted fact is that a syllable is composed of a *nucleus* vowel with or without surrounding consonants which are divided into the *onset* (the consonants preceding the vowel) and the *coda* (the consonants succeeding the vowel). The ONC tagging strategy assigns a tag to every letter of a word based on its role inside the parent syllable. There are three types of tags: O-onset, N-nucleus and C-coda. The numbered ONC makes every tag unique, *inside a syllable*, by adding an index to the tag. To exemplify, we will use the syllabification of the Romanian word "avertisment" (English "warning"). The correct tag sequence for this word is:

$N_1O_1N_1C_1O_1N_1C_1O_1N_1C_1C_2$. Determining where the junctures appear inside the word is easily attained by looking for tag sequences that are unacceptable inside the same syllable such as: C_i-O_j, N_i-N_1, C_i-N_j, N_i-O_j etc. (for whatever indexes i and j). By doing so, we obtain the break sequence: N_1-$O_1N_1C_1$-$O_1N_1C_1$-$O_1N_1C_1C_2$, and with a 1-1 correspondence between tags and letters, we get the sequence "a-ver-tis-ment", which is the correct syllabification of the word.

After iterating through several feature sets we selected the one that yielded the highest results: (l_{-2},l_{-1},l), (l_{-3},l_{-2},l_{-1},l), $(l_{-4},l_{-3},l_{-2},l_{-1},l)$, $(l,l_1,l2)$, (l,l_1,l_2,l_3), (l,l_1,l_2,l_3,l_4), (l_{-1},l,l_1), $(l_{-2},l_{-1},l,l_1,l_2)$, where l is used to mark the current letter and l_i is used to denote the letter at relative distance i from the current one.

3.2 Experiments and results

To test this approach we used a training corpus consisting of 600K syllabified words, compiled from the Romanian Academy Explanatory Dictionary. Using 10-fold validation we obtained and accuracy of **99.01%** on **OOV** words**.** To our knowledge, the best performing system for Romanian syllabification is presented in Ungurean et al. (2011). In their approach, they use Katz-Backoff for determining the most probable n-gram letter split sequence using the output of a stochastic search algorithm. Their method obtained a maximum accuracy of **97.04%** using a window of 5 letter n-grams.

4 Lemmatization

Lemmatization is the process of determining a word's canonical form from its inflectional form. It is a technique useful in various natural language processing applications such as data-mining and document classification. Lemmatization is related to the technique called stemming, which is the process of extracting the longest common subsequence between word forms.

In the case of English, the lemmatization process is fairly simple, but for highly inflectional languages, such as Romanian, this process poses a series of challenges. There are several approaches to this task, with a trend toward rule-based transformations applied to the sequence of characters. The best-performing Romanian lemmatizer [1] (to the best of our knowledge) is implemented after the methodology proposed in Ion (2007). The method builds a lookup table storing for each POS tag (named CTAG), the transformations required for word form to canonical form conversion. When the method has to predict the lemma for a previously unseen word with an associated CTAG (supplied by the POS tagging process), it searches the lookup table for the transformation rules of the CTAG and applies all of them to the unseen word, thus obtaining a set of candidate lemmas from which it probabilistically chooses the most likely one.

4.1 Lemmatization with MIRA

In order to use the MIRA framework, we had to reformulate lemmatization as a sequence labeling task. Our labels are designed to encode the following transformations:

- '*****' – means leave current letter *unchanged*
- '*_nil_*' – means that the current letter must be *removed* from the word's lemma
- '*_r(<character sequence>)* –means that the current letter has to be *replaced* with the character sequence in brackets (<*character sequence*>).

To exemplify, we will use the 2nd person, plural verb "îmbrăcați" (English "dressed"), which has the canonical form "îmbrăca" ("*to* dress"). The letter tag sequence is shown in Table 1.

î	m	b	r	ă	c	a	ț	i
*	*	*	*	*	*	*	_nil_	_nil_

Table 1 - Lemmatization example for word "îmbrăcați"

Lemmatization has to take into account the information provided by the word's morpho-syntactic-description (MSD) tag (Ion, 2007). This means that we either have to train different models for different MSDs or we have to incorporate the MSD information inside the features we use. The Romanian MSDs inventory is very large (more than 600 MSDs) and consequently, the MIRA model obtained by training with MSDs is extremely large, difficult to train and use. Tufiș (1999) presents a strategy for coping with the large Romanian MSD inventory, in which he eliminates lexicon-recoverable morpho-syntactic attributes from the MSDs. The resulting tagset is much smaller and the resulting POS tags are called CTAGs (from Corpus POS tags).

[1] http://ws.racai.ro:9191

In order to reduce our lemmatization model size, we converted every word's MSD from our training set into a CTAG, based on the above mentioned methodology. This reduced our model size about 5 times.

The context used by the labeler is composed of both *lexical* and *morpho-syntactic* features (CTAGs): (l_{-2},l_{-1},l,C), $(l_{-3},l_{-2},l_{-1},l,C)$, $(l_{-4},l_{-3},l_{-2},l_{-1},l,C)$, (l,l_1,l_2,C), (l,l_1,l_2,l_3,C), (l,l_1,l_2,l_3,l_4,C), (l_{-1},l,l_1,C), $(l_{-2},l_{-1},l,l_1,l_2,C)$, where l is used to mark the current letter, l_i is used to denote the letter at relative distance i from the current one and C is used to denote the word form's CTAG.

4.2 Experimental results

Using a training corpus composed of 1M words we withheld 10% for each individual CTAG as the test set. The results of our experiments are shown in Table 1. The overall accuracy of **94%** which is **12%** higher than the results presented in Ion (2007).

In Table 1, all CTAGS beginning with an "N" are nouns, "A" are adjectives and "V" are verbs. The best result (100%) is for invariant adjectives ("A") for which the lemma is the word form. This behavior is preserved for all CTAGs for which lemma is equal to the word form: NSRN (noun, singular, nominative/accusative, non-definite form) with 99.5%, ASN (adjective, singular, non-definite form) with 98.95%, etc. At the opposite pole we find words with CTAGs that are harder to lemmatize: NPN (noun, plural, non-definite form) with 81.51% or NPOY (noun, plural, dative/genitive, definite form) with 83.01% due to their root alternation when going from singular (the number of the lemma) to plural, e.g. for "*stadioan***elor**" (NPOY, English "to the stadiums") lemma is "*stadion*" (English "stadium") where in bold we have the inflectional ending corresponding to the CTAG NPOY and in italic we have the root of the word.

CTAG	# of tokens	# of errors	Accuracy %
A	16	0	100
VN	871	47	94.6
NSON	4223	190	95.5
APOY	5078	99	98.05
NSVN	79	3	96.2
ASN	6205	65	98.95
VPSM	1178	77	93.46
NSOY	6761	279	95.87
ASRY	5121	67	98.69
NP	263	35	86.69
NPRY	6443	884	86.28
VG	2973	118	96.03
NN	263	3	98.86
VPSF	748	15	97.99
APN	6062	127	97.9
NSN	2591	6	99.77
V2	8195	664	91.9
NPOY	6427	1092	83.01
V3	7312	629	91.4
ASON	3030	43	98.58
VPPM	797	58	92.72
NSRY	6701	104	98.45
VPPF	747	15	97.99
V1	6180	455	92.64
APRY	5119	95	98.14
NSRN	4244	19	99.55
ASOY	5122	59	98.85
NPN	6615	1223	81.51
NPVY	28	3	89.29
NSVY	2225	31	98.61
ASVY	626	12	98.08
AN	106	6	94.34
Overall	**112349**	**6523**	**94.19**

Table 2 - Lemmatization results

5 Phonetic transcription

Phonetic transcription (PT; also referred to as grapheme-to-phoneme (G2P) or letter-to-sound (L2S)) can be formalized as finding a relation between letters and corresponding phonemes, which is not a straightforward task and may pose some challenges for languages such as English. For Romanian, phonetic transcription rules are relatively simple compared to English or French (Burileanu, 1999), but there are several exceptions that need to be managed. For the purpose of language independence, data-driven methods are preferable as they only require words and their phonetic transcription equivalents for training, which are easier to obtain than wide coverage set of phonetic transcription rules.

Several Machine Learning (ML) methods have been proposed for the PT task: Black et al. (1998), Jiampojamarn et al. (2008), Pagel et al. (1998), Bisani and Ney (2002), Marchand and Damper (2000) and Demberg (2007).

Jiampojamarn et al. (2008) presented a MIRA based method for L2S conversion of words. Their best result on the English CMU lexicon was 71%. However, the feature template provided in their paper did not turn out to be suitable in our tests. Instead we came up with a different one, which turned out to be the most discriminative for Romanian L2S: (l_{-2},l_{-1},l), (l_{-3},l_{-2},l_{-1},l), $(l_{-4},l_{-3},l_{-2},l_{-1},l)$, (l,l_1,l_2), (l,l_1,l_2,l_3), (l,l_1,l_2,l_3,l_4), (l_{-1},l,l_1), $(l_{-2},l_{-1},l,l_1,l_2)$, (l_{-2},l_{-1},l,l_1), (l_{-1},l,l_1,l_2), where l is used to mark the current letter, l_i is used to denote the letter at relative distance i from the current one.

All the data-driven methods for phonetic transcription require alignments between letters and phonemes. For so-called phonetic (or pseudo-phonetic) languages (e.g. Romanian), the task of grapheme to phoneme conversion is significantly easier and more accurate than for many other languages (such as English). However, there are several issues, common to several languages. The simplest example is that not all words have the same number of phonemes and letters and even if this condition is satisfied, it still does not imply a one-to-one alignment (e.g. experience - IH K S P IH R IY AH N S, where the letter x spawns two phonemes "K" + "S" and the ending "e" is silent; a similar phenomenon happens when we phonetically transcribe the word Romanian "experienţă" (experience) into e k s p e r i e n ts @, where again x spawns "k"+"s"). Expectation-Maximization (EM) can be used to find one-to-one or many-to-many alignments between letters and phonemes (Black et al., 1998; Jiampojamarn et al., 2008; Pagel et al. 1998). Although it is arguable that in the case of Romanian such alignments can be easily attained using simple heuristics, we preferred to use EM on our training data, *to keep our system portable to other languages*.

5.1 Experiments and results

Our training data was extracted from the Romanian Speech Synthesis Corpus (RSS) (Stan et al., 2011) and it is comprised of a small number of words (8K). However, due to the preponderantly phonetical nature of Romanian, this number seems to be sufficient for training a highly accurate L2S data-driven method. Using 10-fold validation we obtained an accuracy of **96.29%** on OOV words, which is comparable to the state-of-the art results (**96.99%**) of a rule-based system reported in Ungurean et al. (2011).

6 Lexical stress prediction

In natural speech certain syllables inside a word have a higher prominence compared to the neighboring syllables of the same word. When this phenomenon occurs, it is said that the syllable is carrying lexical stress. Lexical stress prediction is critical in prosody generation for TTS systems as it governs the correct pronunciation of diverse words and it is used to discriminate between homographs.

6.1 Related work

Oancea and Bădulescu (2003) introduced their rule-based method for lexical stress prediction on Romanian. They trained and tested their method on the same lexicon (4500 words) achieving a **94%** accuracy. Ungurean et al. (2009) used Katz back-off smoothing, for lexical stress assignment based on letter n-grams. Their algorithm works by calculating the probability of every possible combination of stress pattern on an input string. According to their evaluation, this method achieves an accuracy of over **99%** for OOV words.

6.2 Lexical stress prediction with MIRA

Our tagging strategy is inspired after the numbered ONC style encoding used for syllabification. In this case we designed a numbered tagging strategy, in which the "BPS" tag used to label letters which appear before the primary lexical stress; "APS" was used on letters that appear after the primary lexical stress and "PS" to label the letter which carries the primary lexical stress. To exemplify, we will show the labels for the word "îmrăc**a**ţi" (bolded and underlined *a*, receives the primary lexical stress). This type of encoding is available for Romanian, which only uses primary lexical stress. For other languages, which support multiple degrees of lexical stress, the encoding requires adaptations.

î	m	b	r	ă	c	a	ţ	i
BPS	BPS	BPS	BPS	BPS	BPS	PS	APS	APS
1	2	3	4	5	6		1	2

Table 3 – Lexical stress tagging for the word "îmbrăcaţi"

6.3 Experiments and results

Franzén and Horne (1997) conducted a study on stress patterns in Romanian. They showed that stress is rather influenced by derivational affixes

than by inflectional ones, especially for nouns and verbs. Since the vast majority of derivational affixes change the grammatical category of a word, we were motivated to split our training data into 5 categories: nouns (N), verbs (V), adjectives (A), adverbs (R) and mixed (M). This is where the main difference between our approach and other methods can be seen: *splitting the training data based on the part-of-speech increases the overall accuracy by **3.9%*** (see Table 3).

POS	# tokens	# errors	Accuracy
V	11403	42	99.63%
A	11180	55	99.50%
R	52	10	80.77%
N	11060	296	97.32%
Ignored (M)	**33695**	**1718**	**94.90%**
Overall	**33695**	**403**	**98.80%**

Table 4 - Lexical stress accuracy

When predicting the primary lexical stress position for a given word, a model is chosen based on the POS tag of the given word. If the POS is different from the first four categories or if it is unknown (if there is no context available), the system uses the *mixed model*, which is a model created by training on the entire lexicon regardless of the POS.

The lexical feature templates we used for lexical stress prediction are identical to the ones we used for lemmatization.

7 Conclusions

In this paper we addressed the task of lexical processing for OOV words, which are one of the main sources of errors in both speech synthesis and natural language processing applications. We presented a unified data-driven framework that is designed to accurately handle the lemmatization, syllabification, phonetic transcription and lexical stress prediction of *OOV words*. Although, our main focus was on Romanian, the advantage of using data-driven methods is that with proper training lexicons and, in some cases, with minor adjustments, they *can be applied to any other language*.

Our results are better than state-of-the-art results cited for Romanian in the case of syllabification (99% vs. 97%) and lemmatization (94% vs. 82%), and only slightly worse for phonetic transcription (96.3% vs. 97%) and lexical stress prediction (98.8% vs. 99%), which can be explained by the fact that we did not incorporate any explicit knowledge of Romanian into our algorithms. In this context, we should emphasize that we successfully employed the MIRA framework described in this paper (without any modifications) to do phonetic transcription for English, French, German and Dutch and lemmatization for Serbian with very good results.

The methods we presented are already implemented in a natural language pre-processing tool written entirely in JAVA for portability and available as an open-source package.

Acknowledgments

The work reported here was funded by the project METANET4U by the European Commission under the Grant Agreement No 270893

References

Bartlett, S., Kondrak, G., & Cherry, C. 2008. Automatic syllabification with structured SVMs for letter-to-phoneme conversion. *Proceedings of ACL-08: HLT*, 568-576.

Black, A. W., Lenzo, K., & Pagel, V. 1998. Issues in building general letter to sound rules. In *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis*.

Burileanu, D., Sima, M., & Neagu, A. 1999. A phonetic converter for speech synthesis in Romanian. In *Proc. of the XIVth International Congress on Phonetic Sciences ICPhS'99* (pp. 503-506).

Crammer, K. and Singer, Y. 2003. Ultraconservative online algorithms for multiclass problems. The Journal of Machine Learning Research, 3:951–991.

Daelemans, W., Van Den Bosch, A., & Weijters, T. 1997. IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11(1), 407-423.

Demberg, V., Schmid, H., & Mohler, G. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Annual Meeting-Association for Computational Linguistics* (Vol. 45, No. 1, p. 96).

Franzén, V., & Horne, M. 2009. Word stress in Romanian. *Lund Working Papers in Linguistics*, *46*, 75-91.

Ion, R. 2007. Word Sense Disambiguation Methods Applied to English and Romanian. PhD thesis (in Romanian). Romanian Academy, Bucharest.

Jiampojamarn, S., Cherry, C., & Kondrak, G. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. Proceedings *of ACL-08: HLT*, 905-913.

Kahn, D. 1976. Syllable-based generalizations in English phonology (Vol. 156). Bloomington: Indiana University Linguistics Club.

Lafferty, J., McCallum, A., & Pereira, F. C. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Oancea, E., & Badulescu, A. 2002. Stressed syllable determination for Romanian words within speech synthesis applications. *International Journal of Speech Technology*, *5*(3), 237-246.

Stan, A., Yamagishi, J., King, S., & Aylett, M. 2011. The Romanian Speech Synthesis (RSS) corpus: building a high quality HMM-based speech synthesis system using a high sampling rate. *Speech Communication*, 53(3), 442-450.

Tufiş, D. 1999. Tiered tagging and combined language models classifiers. *Text, Speech and Dialogue* (pp. 843-843). Springer Berlin/Heidelberg.

Ungurean, C., Burileanu, D., Popescu, V. and Derviş, A. 2011. Hybrid Syllabification and Letter-To-Phone Conversion For TTS Synthesis. In *U.P.B. Sci. Bull.*, Series C, Vol. 73, Iss. 3, 2011, ISSN 1454-234x

Weijters, A. 1991. A simple look-up procedure superior to NETtalk?. In *Proceedings of the International Conference on Artificial Neural Networks - ICANN-91*, Espoo, Finland