Vowel and Diacritic Restoration for Social Media Texts

Kübra Adalı
Dep. of Computer Eng.
Istanbul Technical University
Istanbul, Turkey
kubraadali@itu.edu.tr

Gülşen Eryiğit
Dep. of Computer Eng.
Istanbul Technical University
Istanbul, Turkey
gulsen.cebiroglu@itu.edu.tr

Abstract

In this paper, we focus on two important problems of social media text normalization, namely: vowel and diacritic restoration. For these two problems, we propose a hybrid model consisting both a discriminative sequence classifier and a language validator in order to select one of the morphologically valid outputs of the first stage. The proposed model is language independent and has no need for manual annotation of the training data. We measured the performance both on synthetic data specifically produced for these two problems and on real social media data. Our model (with 97.06% on synthetic data) improves the state of the art results for diacritization of Turkish by 3.65 percentage points on ambiguous cases and for the vowel restoration by 45.77 percentage points over a rule based baseline with 62.66% accuracy. The results on real data are 95.43% and 69.56% accordingly.

1 Introduction

In recent years, with the high usage of computers and social networks like Facebook and Twitter, the analysis of the social media language has become a very popular and crucial form of business intelligence. But unfortunately, this language is very different from the well edited written texts and much more similar to the spoken language, so that, the available NLP tools do not perform well on this new platform.

As we all know, Twitter announced (at April 1st, 2013)[1] that it is shifting to a two-tiered service where the basic free service 'Twttr" will only allow to use consonants in the tweets. Although, this is a very funny joke, people nowadays are already very used to use this style of writing without vowels in order to fit their messages into 140 characters Twitter or 160 characters SMS messages. As a result, the vowelization problem (Twttr ⇒ Twitter) is no more limited with some specific language families (e.g.semitic languages) (Gal, 2002; Zitouni et al., 2006) but it became a problem of social media text normalization in general.

Diacritics are some marks (e.g. accents, dots, curves) added to the characters and have a wide usage in many languages. The absence of these marks in Web2.0 language is very common and posses a big problem in the automatic processing of this data by NLP tools. Although, in the literature, the term "diacritization" is used both for vowel and diacritic restoration for semitic languages, in this paper, we use this term only for the task of converting an ASCII text to its proper form (with accents and special characters). A Turkish example is the word "dondu" (*it is frozen*) which may be the ascii form of both "dondu"(*it is frozen*) or "döndü" (*it returned*) where the ambiguity should be resolved according to the context. In some studies, this task is also referred as "unicodification"(Scannell, 2011) or "deasciification"(Tür, 2000).

In this paper, we focus on these two important problems of social text normalization, namely: diacritization and vowelization. These two problems compose almost the quarter (26.5%) of the normalization errors within a 25K Turkish Tweeter Data Set. We propose a two stage hybrid model: firstly a discriminative model as a sequence classification task by using CRFs (Conditional Random Fields) and secondly a language validator over the first stage's results. Although in this paper, we presented our results on Turkish (which is a highly agglutinative language with very long words full of un-ascii characters), the proposed model is totally language independent and has no need for manual

[1] https://blog.twitter.com/2013/annncng-twttr

annotation of the training data. For morphologically simpler languages, it would be enough to use a lexicon lookup for the language validation stage (whereas we used a morphological analyzer for the case of Turkish). With our proposed model, we obtained the highest results in the literature for Turkish diacritization and vowelization.

The remaining of the paper is structured as follows: Section 2 discusses the related work, Section 3 tries to show the complexity of diacritization and the vowelization tasks by giving examples from an agglutinative language; Turkish. Section 4 introduces our proposed model and Section 5 presents our experiments and results. The conclusion is given in Section 6.

2 Related Work

The vowelization problem is mostly studied for semitic languages and many different methods are applied to this problem. The problem is generally referred as diacritization for these languages, since diacritics are placed on consonants for the purpose of vowel restoration. For example, the short vowels in Arabic are only pronounced by the use of diacritics put on other consonants. Some of these studies are as follows: Gal (2002) reports the results on Hebrew by using HMMs and Zitouni et al. (2006) on Arabic by using maximum entropy based models. Al-Shareef and Hain (Al-Shareef and Hain, 2012) deals with the vowelization of colloquial Arabic for automatic speech recognition task by using CRFs on speaker and contextual information. Haertel et al. (2010) uses conditional markov models for the vowel restoration problem of Syriac. Nelken and Shieber (2005) uses a finite state transducer approach for Arabic as well. To the best of our knowledge, the vowelization work on Turkish is the first study on a language which do not possess the vowelization problem by its nature. We believe that on that sense, our hybrid model will be a good reference for future studies in social media text normalization where the problem is disregarded in recent studies.

The diacritization task on the other hand is not addressed as frequently as the vowelization problem[2]. Some studies are as follows: Scannell (2011) uses a Naive Bayes classifier for both word-level and character-level modeling. Each ambiguous character in the input is regarded as an independent classification problem. They are using lexicon lookup which is not feasible for every possible word surface form in agglutinative or highly inflected languages. They refer to a language model in ambiguous cases. They tested their system for 115 languages as well as for Turkish (92.8% on a much easier data set than ours (refer to Section 5.1) . Simard and Deslauriers (2001) tries to recover the missing accents in French. They are using a generative statistical model for this purpose. De Pauw et al. (2007) also test their MBL (memory based learning) model on different languages. Although they do not test for Turkish, the most attractive part of theirs results is that the performances for highly inflectional languages differ sharply from the others towards the negative side. Nguyen and Ock (2010) deals with the diacritization of Vietnamese by using Adaboost and C4.5.

The work done so far for the diacritization of Turkish are from Tür (2000) (character-based HMM model), Zemberek (2007), Yüret and de la Maza (Yüret and de la Maza, 2006) (GPA: a kind of decision list algorithms). We give the comparison of the two later systems on our data set and propose a discriminative equivalent of the HMM approach used in Tür (2000) (see Section 5 for further discussions). For the vowelization, the only study that we could find is from Tür (2000) which uses again the same character-level HMM model into this problem (with an equivalent discriminative model given at Table 8 $\pm$3ch model).

3 The complexity

This section tries to draw light upon the complexity of diacritization and the vowelization tasks by giving examples from an agglutinative language; Turkish.

3.1 Turkish

Turkish is an agglutinative language which means that words have theoretically an infinite number of possible surface forms due to the iterative concatenation of inflectional and derivational suffixes. As for other similar languages, this property of the language makes impractical for Turkish words to be validated by using a lexicon. And also, the increasing length[3] of the words creates a big search space especially for the vowelization task.

[2]Here, we exclude all the works done for semitic languages. The reason is explained on the former paragraph.

[3]The average word length is calculated as 6.1 for Turkish nouns and 7.6 for verbs in a 5 million word corpus(Akın and Akın, 2007).

Turkish alphabet has 7 non-ascii characters that don't exist in the Latin alphabet (ç, ğ, ı, İ, ö, ş, ü) and the ascii counterparts of these letters (c, g, i, I, o, s, u) are also valid letters in the alphabet which causes an important disambiguation problem at both word and sentence level. The alphabet contains 8 vowels (a(A), e(E), ı(I), i(İ), o(O), ö(Ö), u(U), ü(Ü)) in total.

3.2 Diacritization

The following real example sentence taken from social media "Ruyamda evde oldugunu gordum.", written by using only the ascii alphabet, has two possible valid diacritized versions:

1. "Rüyamda evde olduğunu gördüm."
(*I had a dream that you were at home.*)
2. "Rüyamda evde öldüğünü gördüm."
(*I had a dream that you died at home.*)

As can be observed from this sentence some of the asciified words (e.g. "oldugunu") has more than one possible valid counterparts which causes the meaning change of the whole sentence.

The problem is the decision of the appropriate forms for the critical letters (C, G, I, O, S, U)[4]. Although the problem seems like a multi-class classification problem, it is in essence a binary-classification task for each critical letter and can be viewed as a binary sequence classification task for the whole word so that the original word will be chosen from (2^n) possibilities where n is the number of occurrence of critical letters (C, G, I, O, S, U) in the ascii version. For example the word "OldUGUnU" has (2^5=32) possible transformations whereas only 2 of them ("olduğunu" and "öldüğünü") are valid Turkish words. Figure 1 gives a second example and shows all the possible (2^2=4) diacritized versions of the word "aCI" where again only two of them are valid words (emphasized with a bold background colour): "açı"(*angle*) and "acı"(*pain*).

3.3 Vowelization

Vowelization on the other hand causes much more complexity when compared to diacritization. Each position[5] between consequent consonants, at the beginning or ending of the word may take one vowel or not resulting a selection from 9 class labels (the 8 vowel letters + the null character). For example, the vowelization of the word "slm"("*hi*" written without vowels, with n=4 positions _s_l_m_) will produce 9^4 = 6561 possibilities where 39 of them are valid Turkish words (e.g. "salam"(*salami*), "sulama"(*watering*), "salım"(*my raft*), "selam"(*hi*), "sılam"(*my furlough*) etc...).

[4]From this point on, we will show the ascii versions of these letters as capital letters meaning that they may appear in the diacritized version of the word either in their ascii form or in their diacritized form. Ex: the capital C will become either c or ç after the processing.

[5]For the sake of simplicity, we just assumed that only zero or one vowel may appear between two consonants whereas there exist some words with consecutive vowels (such as "saat"(*clock*)) although very rarely

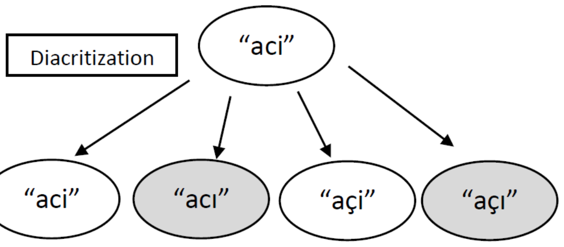

Figure 1: Possible Diacritized Versions of "aCI"

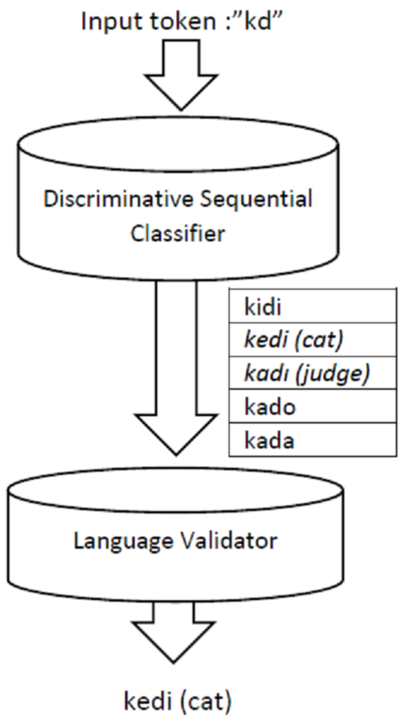

Figure 2: Proposed Model

4 Proposed Model

Most of the previous work in the literature (Section 2) uses either some (generative of discriminative) machine learning models or some nlp tools (e.g. morphological analyzers, pos taggers, linguistic rules) in order to solve the vowelization problem. As it is shown in the previous section, for languages with higher number of vowels and word length due to their rich inflectional morphology, the search space gets very high very rapidly. Since the problem is mostly similar to generation, in order to increase the likelihood of the generated output word, most of the approaches include character level probabilities or relationships. In this case, it is unfair to expect from a machine learning system to generate morphologically valid outputs (especially for highly inflectional languages) while trying to maximize the overall character sequence probability.

We propose a two stage model (Figure 2) which has basically two components.

1. a discriminative sequence classifier
2. a language validator

4.1 Discriminative Sequence Classifier

In the first stage, we use CRFs[6] (Lafferty et al., 2001) in order to produce the most probable output words. This stage treats the diacritization and vowelization as character level sequence labeling tasks, but since it is a discriminative model, it is also possible to provide neighboring words as features into the system. During training, each instance within a sequence has basically the following main parts:

1. features related to the current and neighboring tokens (namely surface form or lemma)
2. features related to the current and neighboring characters[8]
3. class label

The test data is also prepared similarly except the gold standard class labels.

Table 1 and Table 2 show instance samples for the sample words (“OldUGUnU” and “_s_l_m_ ”) given in Section 3. As can be observed from the tables, we have 7 different class labels in diacritic restoration and 9 different class labels in vowel restoration (one can refer to Section 3 for the details). The sequences represent words in focus and each instance line within a sequence stands for the character position in focus. The sample for diacritization has 5 character features and 2 word features where the current character feature limits the number of the class labels to be assigned to that position by 2. The sample for vowelization has 1 word feature and 6 character features.

[6]In this work, we used CRF++[7] which is an open source implementation of CRFs.

[8]The feature related to the current character is only available in diacritization model

Curr. Letter	Neig. Word(+1)	Curr. Word	Neig. Ch(-2)	Neig. Ch(-1)	Neig. Ch(+1)	Neig. Ch(+2)	Class Label
O	GOrdUm	OldUGUnU	_	_	l	d	ö
U	GOrdUm	OldUGUnU	l	d	G	U	ü
G	GOrdUm	OldUGUnU	d	U	U	n	ğ
U	GOrdUm	OldUGUnU	U	G	n	U	ü
U	GOrdUm	OldUGUnU	U	n	_	_	ü

Table 1: Diacritization: Instance Representation for the word ”oldugunu”
“OldUGUnU” 5 critical positions

Curr. Word	Neig. Ch(-3)	Neig. Ch(-2)	Neig. Ch(-1)	Neig. Ch(+1)	Neig. Ch(+2)	Neig. Ch(+3)	Class Label
slm	_	_	_	s	l	m	_
slm	_	_	s	l	m	_	e
slm	_	s	l	m	_	_	a
slm	s	l	m	_	_	_	_

Table 2: Vowelization: Instance Representation for the word “slm”
“_s_l_m_” 4 possible vowel positions

CRFs are log-linear models and in order to get advantage of the useful feature combinations, one needs to provide these as new features to the CRFs. In order to adopt a systematic way, we took the features’ combinations for character features and word features separately. For character features we took the combinations up to 6-grams for $\pm$3ch and for the neighboring word features up to 4 grams. The number of features affects directly the maximum amount of training data that could be used during the experiments. The total number of feature templates after the addition of feature combinations ranges between 7 for the simplest case and 30 for our model with maximum number of features. Three sample feature templates are given below for the sample sequence of Table 1. The templates are given in [pos,col] format, where pos stands for the relative position of the token in focus and col stands for the feature column number in the input file. U06 is the template for using the sixth[9] feature in Table 1 (Neigh. Ch(+1)). U13 is a bigram feature combination of 2nd and 3th features (the current token and the next token). U11 is a fourgram feature combination of 4th, 5th, 6th and 7th features of our feature set that refers to the group of the previous two characters and the next two characters.

[9]in CRF++ feature templates the features indexes start from 0.

$$U06 : \%x[0,5]$$
$$U13 : \%x[0,1]/\%x[0,2]$$
$$U11 : \%x[0,3]/\%x[0,4]/\%x[0,5]/\%x[0,6]$$

4.2 Language Validator

The n best sequences of the discriminative classifier is then transferred to the language validator. We use a two-level morphological analyzer (Şahin et al., 2013) for the Turkish case since in this agglutinative language it is impractical to validate a word by making a lexicon lookup. But this second part may be replaced by any language validator (for other languages) which will filter only the valid outputs from the n best results of the discriminative classifier. Figure 2 shows an example case of the process for vowelization. The system takes the consonant sequence "kd" and the 5 best output of the first stage is produced as "kidi, kedi, kadı, kado, kada". The language validator then chooses the most probable valid word "kedi" (*cat*) as its output. One should notice that if none of the n most probable results is a valid word, then the system won't produce any suggestion at all. We show experimental results on the effect of the selection of n in the next section.

5 Experimental Setup And Results

In this section, we first present our datasets and evaluation strategy. We then continue with the diacritization experiments and finally we share the results of our vowelization experiments.

5.1 Datasets and Evaluation Methodology

For both of the diacritization and vowelization tasks, creating the labeled data is a straightforward task since the reverse operations for these (converting from formally written text to their Ascii form or to a form without vowels) can be accomplished automatically for most of the languages (except semitic languages where the vowels do not appear in the formal form). To give an example from Turkish, the word "olduğunu" may be automatically converted to the form "OldUGUnU" for diacritization and "_l_d_g_n_" for vowelization experiments. We used data from three different corpora: METU Corpus (Say et al., 2002) and two web corpora from Yıldız and Tantuğ (2012) and Sak et al. (2011).

In order to judge different approaches fairly, we aimed to create a decently though test set. Since the vowelization task already comprises a very high ambiguity, we focused to the ambiguous diacritization samples. With this purpose, we first took the Turkish dictionary and converted all the lemmas within the dictionary into their Ascii forms. We then created the possible diacritized forms (Figure 1) and created a list of all ambiguous lemmas (1221 lemmas in total) by finding all the lemmas which could be produced as the output of diacritization. For example "açı" and "acı" are put into this list after this operation. Although this ambiguous lemmas list may be extended by also considering interfusing surface forms, for the sake of simplicity we just considered to take the ambiguous lemmas from the dictionary. We then searched our three corpora (and the WEB where not available in these) for the words with an ambiguous lemma and created our test set so that for each ambiguous lemma there is exactly one sentence consisting of it. As a result, we collected a test set of 1157 sentences (17923 tokens) consisting of 1871 ambiguous words[10] in total. The remaining sentences from the corpora are used during training. Since the feature set size directly affects the amount of usable training data, for different experiment sets we used different size of training data each time trying to use the data from the three corpora in equal amounts.

After evaluating with synthetically produced training and test sets, we also tested our best performed models on real data collected from social media (25K tweets with at least one erroneous token) and normalized manually (Eryiğit et al., 2013). This data consists 58836 tokens that have text normalization problems where 3.75% is due to missing vowels and 22.8% is due to misuse of Turkish characters. In order to separate these specific error sets, we first automatically aligned the original and normalized tweets and then applied some filters over the aligned data: e.g. Deascification errors are selected so that the character length of the original word and its normalized form should be the same and the differing letters should only be the versions of the same Turkish characters.

For the evaluation of diacritization, we provide two accuracy scores: Accuracy over the entire words ($Acc_{Overall}$ Equation 1) and accuracy over the ambiguous words alone (Acc_{Amb} Equation 2

[10]One should note that each sentence in the test set contains at least one or more ambiguous surface forms. The test data will be available to the researches via http://...

over 1871 ambiguous words in the test set). Since the vowelization problem is almost entirely ambiguous, the two scores are almost the same for the entire tests ($\# \ of \ words \approx \# \ of \ amb. \ words$). That is why, for the vowelization task we provide only $Acc_{Overall}$.

$$Acc_{Overall} = \frac{\# \ of \ corr. \ diacritized \ words}{\# \ of \ words} \tag{1}$$

$$Acc_{Amb} = \frac{\# \ of \ corr. \ diacritized \ amb. \ words}{\# \ of \ amb. \ words} \tag{2}$$

In the work of Tür (2000), the accuracy score is provided as the correctly determined characters which we do not find useful for the given tasks: $Acc_{Amb} = \frac{\# \ of \ corr. \ diacritized \ amb. \ chars}{\# \ of \ amb. \ chars}$. This score gives credit to the systems although the produced output word is not a valid Turkish word. For example, if a vowelization system produces an invalid output as "oldgn" for the input "_l_d_g_n_", it will have a 1/5 (one correct character over 5 possible positions) score whereas in our evaluation this output will be totally penalized.

5.2 Diacritization Experiments

For diacritization, we designed four sets of experiments. The first set of experiments (Table 3) presents the results of our baseline systems. We provide four baseline systems. The first one is a rule based diacritics restorer which creates all the possible diacritics for a given input and outputs the first morphologically valid one. As the proposed model does, the rule based system also validates its outputs by using the morphological analyzer introduced in Section 4.2. One can see from the table that the accuracy on the ambiguous words of this system is nearly 70%. Our second baseline uses a unigram language model in order to select the most probable valid output of the morphological analyzer. Our third baseline is a baseline for our discriminative classifier (with only ±2 neighboring characters) without the language validator component. In this model, the top most output of the CRF is accepted as the correct output word. One can observe that this baseline although it performs better than the rule based system, it is worse than the second baseline with a language model component. Our last baseline is the baseline for the proposed system in this paper with a discriminative classifier (using only ±2 neighboring characters) and a language validator which chooses the first valid output within the top 5 results of the classifier. It outperforms all the previous baselines.

System	Acc Overall	Acc Amb
Rule based	90.38	69.17
Rule based + Unigram LM	91.94	83.54
CRF $\pm 2ch$	87.93	77.24
CRF ±2ch + Lang.Valid.	**94.88**	**88.51**

Table 3: Diacritization Baseline Results

The second set of experiments given in Table 4 is for the feature selection of the proposed model. We test with the neighboring characters up to ±3 and together with the surface form of the current token $sform_{curr}$ and/or the first n characters of the current token $firstnch_{curr}$ as lemma feature. For both of the first two sets of experiments (Table 3 and Table 4) we used a training data of size 4591K (the max. possible size for the most complex feature set in these experiments; (last line of Table 4). It can be observed from Table 4 that although ±3ch (2nd line) performs better than ±2ch (1st line), when we use these together with $sform_{curr}$ we obtain better results with ±2ch (3rd line). Since ±3ch (7 characters in total) will be very close to the whole number of characters within the surface form, the new feature's help is more modest in ±3ch model. In these experiments we try to optimize on the overall accuracies. Our highest score in this table is with the $\pm 2ch + sform_{curr} + first5ch_{curr}$ (last line) but since the difference between this and $\pm 2ch + sform_{curr}$ is not statistically significant (with McNemar's test p<0.01) and the size of the maximum possible training data could still be improved for the latter model, we decided to continue with $\pm 2ch + sform_{curr}$.

In the third set of diacritization experiments (Table 5) we investigated the effect of using the neighboring tokens as features. In this experiment set, the training data size is decreased to a much lower size, only 971K in order to be able to train with ±2 neighboring tokens. Each line of the table is the addition of the surface forms for the precised positions to the model of the first line $\pm 2ch + sform_{curr}$. We tested with all the combinations in the ±2 window size. For exam-

Feature Combinations	Acc Overall	Acc Amb
$\pm 2ch$	94.88	88.51
$\pm 3ch$	95.76	91.05
$\mathbf{\pm 2ch + sform_{curr}}$	**96.26**	**91.60**
$\pm 3ch + sform_{curr}$	96.20	91.71
$\pm 2ch + first3ch_{curr}$	95.29	90.17
$\pm 2ch + first4ch_{curr}$	95.60	89.06
$\pm 2ch + first5ch_{curr}$	95.95	90.72
$\pm 2ch + sform_{curr} + first3ch_{curr}$	96.23	91.82
$\pm 2ch + sform_{curr} + first4ch_{curr}$	96.26	91.82
$\pm 2ch + sform_{curr} + first5ch_{curr}$	96.28	91.60

Table 4: Diacritization Feature Selection I

Features	Acc Overall	Acc Amb
$\pm 2ch + sform_{curr}$	95.29	90.61
$\mathbf{+sform_{0010}}$	**95.49**	**90.72**
$+sform_{0011}$	95.39	90.39
$+sform_{0100}$	93.77	83.32
$+sform_{0110}$	95.39	90.28
$+sform_{0111}$	95.26	89.95
$+sform_{1100}$	95.24	89.83
$+sform_{1110}$	95.21	89.50
$+sform_{1111}$	95.11	89.17

Table 5: Diacritization Feature Selection II

ple $sform_{0010}$ means that the surface form of the token at position +1 is added to the features. This feature set outperformed all the other ones.

System	Acc Overall	Acc Amb
Yüret (2006)	95.93	91.05
Zemberek (2007)	87.71	82.55
$\pm 2ch + sform_{curr}$	96.15	92.04
$\mathbf{\pm 2ch + sform_{curr} + sform_{0010}}$	**97.06**	**94.70**

Table 6: Diacritization Results Comparison with Previous Work

Finally, in Table 6, we give the comparison results of our proposed model with the available Turkish deasciifiers (the tools' original name given by the authors) (Yüret and de la Maza, 2006; Akın and Akın, 2007). We both tested by $\pm 2ch + sform_{curr}$ and $\pm 2ch + sform_{curr} + sform_{0010}$. Both of the models are tested with maximum possible size of training data: 10379K and 5764K successively. Our proposed model for diacritization outperformed all of the other methods with a success rate of 97.06%. It outperformed the state of the art by 1.13 percentage points in overall accuracy and by 3.65 percentage points in ambiguous cases (both results statistically significant).

5.3 Vowelization Experiments

For the vowelization, we designed similar set of experiments. In Table 7, we provide the results for a rulebased baseline and our proposed model with $\pm 2ch$. It is certainly a very time consuming process to produce all the possible forms for the vowelization task (see Section 3.3). Thus, for the rule based baseline we stopped the generation process once we find a valid output. The baseline of the proposed model provides a 28.44 percentage points improvements over the rule based system. We did not try to compare our results with the work of Tür (2000) (an HMM model on character level) firstly because the developed model was not available for testing, secondly because the provided evaluation (see Section 5.1) was useless for our purposes and finally because our ± 3 character model provided in the second line of Table 8 is a discriminative counterpart of his 6-gram generative model.

System	Acc Overall
Rule based	16.89
CRF ±2ch+Lang.Valid.	**45.33**

Table 7: Vowelization Baseline Results

Table 8 gives the feature selection tests' results similarly to the previous section. This time we obtained the highest score with $\pm 3ch + sform_{curr}$ 59.17%. In this set of experiments, we used 4445K of training data.

In order to investigate the impact of neighboring tokens, in the experiments given in Table 9, we had to continue with $\pm 2ch + sform_{curr}$ with

Feature Combinations	Acc Overall
$\pm 2ch$	45.33
$\pm 3ch$	57.20
$\pm 2ch + sform_{curr}$	57.22
$\mathbf{\pm 3ch + sform_{curr}}$	**59.17**
$\pm 2ch + first3ch_{curr}$	40.44
$\pm 2ch + first4ch_{curr}$	40.48
$\pm 2ch + first5ch_{curr}$	44.22
$\pm 2ch + sform_{curr} + first3ch_{curr}$	45.89
$\pm 2ch + sform_{curr} + first4ch_{curr}$	45.89
$\pm 2ch + sform_{curr} + first5ch_{curr}$	49.58

Table 8: Vowelization Feature Selection I

Features	Acc Overall
$\pm\mathbf{2ch} + \mathbf{sform_{curr}}$	**54.07**
$+sform_{0010}$	50.89
$+sform_{0011}$	49.60
$+sform_{0100}$	31.84
$+sform_{0110}$	49.41
$+sform_{0111}$	47.78
$+sform_{1100}$	48.98
$+sform_{1110}$	47.88
$+sform_{1111}$	47.21

Table 9: Vowelization Feature Selection II

971K of training data.[11] We could not obtain any improvement with the neighboring tokens. We relate these results to the fact that the neighboring tokens are also in vowel-less form in the training data so that this information do not help the disambiguation of the current token. Since we could not add the word based features to this task by this model, for future work we are planning to apply a word based language model over the proposed model's possible output sequences.

In the final experiment set given in Table 10, we trained our best performing model $\pm 3ch + sform_{curr}$ with the maximum possible training data (6653K). We also tested with different N values of CRF output. Although there is a slight increase on the overall accuracy by passing from N=5 to N=10, the increase is much higher when we evaluate with Acc_{topN}. Equation 3 gives the calculation of this score which basically calculates the highest score that could be obtained after perfect reranking of the top N results. In this score the system is credited if the correct vowelized answer is within the top N results of the system. We see from the table that there is still a margin for the improvement in top 10 results (up to 85.09% for the best model). This strengthens our believe for the need of a word based language model over the proposed model outputs. Our vowelization model in its current state achieves an accuracy score of 62.66% with a 45.77 percentage points improvements over the rule based baseline.

$$Acc_{topN} = \frac{\sum 1\ if\ result\ exists\ within\ top\ N}{\#\ of\ words} \tag{3}$$

System	Acc Overall	Acc top N
$\pm 3ch + sform_{curr}$ With Top 5 Poss. from CRF	62.05	80.21
$\pm 3ch + sform_{curr}$ With Top 7 Poss. from CRF	62.36	82.53
$\pm 3ch + sform_{curr}$ With Top 10 Poss. from CRF	**62.66**	**85.09**

Table 10: Vowelization Top N Results

Finally we test our best models on vowelization and diacritization errors from our Tweeter data set and obtained 95.43% for diacritization and 69.56% for vowelization.

6 Conclusion And Future Work

In this paper, we proposed a hybrid model for the diacritization and vowelization tasks which is an emerging problem of social media text normalization. Although the tasks are previously investigated for different purposes especially for semitic languages, to the best of our knowledge, this is the first time that they are evaluated together for the social media data on a language which do not possess these problems in its formal form but only in social media platform. We obtained the highest scores for the diacritization (97.06%) and vowelization (62.66%) of Turkish.

We have two future plans for the vowelization part of the proposed model. The first one, as detailed in previous section, is the application of a word based language model over the valid CRF outputs. The second one is the extension for partial vowelization. Although in this work, we designed the vowelization task as the overall generation of the entire vowels within a vowel-less word, we observe from the social web data that people also tend to write with partially missing vowels. As an example, they are writing "sevyrm" instead of the word "seviyorum" (*I love*). In this case, the position between the consonants 's' and 'v' is constrained to the letter 'e' and it is meaningless to generate the other possibilities for the remaining 7 vowels. For this task, we are planning to focus on constrained Viterbi algorithms during the decoding stage.

[11]If we select the larger model, it is going to be impossible to feed enough training data to the system. Since in this set of experiments (Table 9) we only investigate the impact of neighboring tokens, we had/preferred to select the smaller character model.

The tool's web api and the produced data sets will be available to the researchers from the following address http://tools.nlp.itu.edu.tr/(Eryiğit, 2014)

Acknowledgment

This work is part of a research project supported by TUBITAK 1001(Grant number: 112E276) as an ICT cost action (IC1207) project.

References

Ahmet Afsin Akın and Mehmet Dündar Akın. 2007. Zemberek, an open source nlp framework for turkic languages. *Structure*.

Sarah Al-Shareef and Thomas Hain. 2012. Crf-based diacritisation of colloquial Arabic for automatic speech recognition. In *INTERSPEECH*. ISCA.

Guy De Pauw, Peter W. Wagacha, and Gilles-Maurice de Schryver. 2007. Automatic diacritic restoration for resource-scarce languages. In *Proceedings of the 10th international conference on Text, speech and dialogue*, TSD'07, pages 170–179, Berlin, Heidelberg. Springer-Verlag.

Gülşen Eryiğit, Fatih Samet Çetin, Meltem Yanık, Tanel Temel, and İyas Çiçekli. 2013. Turksent: A sentiment annotation tool for social media. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 131–134, Sofia, Bulgaria, August. Association for Computational Linguistics.

Gülşen Eryiğit. 2014. ITU Turkish NLP web service. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden, April. Association for Computational Linguistics.

Ya'akov Gal. 2002. An hmm approach to vowel restoration in Arabic and Hebrew. In *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, SEMITIC '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robbie A. Haertel, Peter McClanahan, and Eric K. Ringger. 2010. Automatic diacritization for low-resource languages using a hybrid word and consonant cmm. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 519–527, Stroudsburg, PA, USA. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

Rani Nelken and Stuart M. Shieber. 2005. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Semitic '05, pages 79–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kiem-Hieu Nguyen and Cheol-Young Ock. 2010. Diacritics restoration in Vietnamese: letter based vs. syllable based model. In *Proceedings of the 11th Pacific Rim international conference on Trends in artificial intelligence*, PRICAI'10, pages 631–636, Berlin, Heidelberg. Springer-Verlag.

Muhammet Şahin, Umut Sulubacak, and Gülşen Eryiğit. 2013. Redefinition of Turkish morphology using flag diacritics. In *Proceedings of The Tenth Symposium on Natural Language Processing (SNLP-2013)*, Phuket, Thailand, October.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2011. Resources for Turkish morphological processing. *Lang. Resour. Eval.*, 45(2):249–261, May.

Bilge Say, Deniz Zeyrek, Kemal Oflazer, and Umut Özge. 2002. Development of a corpus and a treebank for present-day written Turkish. In *Proceedings of the Eleventh International Conference of Turkish Linguistics*, Famaguste, Cyprus, August.

Kevin P. Scannell. 2011. Statistical unicodification of African languages. *Lang. Resour. Eval.*, 45(3):375–386, September.

Michel Simard and Alexandre Deslauriers. 2001. Real-time automatic insertion of accents in French text. *Nat. Lang. Eng.*, 7(2):143–165, June.

Gökhan Tür. 2000. *A statistical information extraction system for Turkish.* Ph.D. thesis, Department of Computer Engineering and the Institute of Engineering and Science of Bilkent University, Ankara.

Eray Yıldız and Cüneyd Tantuğ. 2012. Evaluation of sentence alignment methods for English-Turkish parallel texts. In *Proceedings of the First Workshop on Language Resources and Technologies for Turkic Languages (LREC)*, Istanbul, Turkey, 23-25 May.

Deniz Yüret and Michael de la Maza. 2006. The greedy prepend algorithm for decision list induction. In *Proceedings of the 21st international conference on Computer and Information Sciences*, ISCIS'06, pages 37–46, Berlin, Heidelberg. Springer-Verlag.

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of Arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 577–584, Stroudsburg, PA, USA. Association for Computational Linguistics.