Training Automatic Transliteration Models on DBpedia Data

Velislava Todorova
Linguistic Modeling Department
IICT-BAS
`slava@bultreebank.org`

Kiril Simov
Linguistic Modeling Department
IICT-BAS
`kivs@bultreebank.org`

Abstract

Our goal is to facilitate named entity recognition in Bulgarian texts by extending the coverage of DBpedia (`http://www.dbpedia.org/`) for Bulgarian. For this task we have trained translation Moses models to transliterate foreign names to Bulgarian. The training sets were obtained by extracting the names of all people, places and organizations from DBpedia and its extension Airpedia (`http://www.airpedia.org/`). Our approach is extendable to other languages with small DBpedia coverage.

1 Introduction

DBpedia Linked Open Dataset[1] provides the wealth of Wikipedia in a formalized way via the ontological language in which DBpedia statements are represented. Still DBpedia reflects the multilingual nature of WikiPedia. But if a user needs to access the huge number of instances extracted from the English version of WikiPedia, for example, in a different language (in our case Bulgarian) he/she will not be able to do so, because the DBpedia in the other language will not provide appropriate Uniform Resource Identifiers (URIs) for many of the instances in the English DBpedia. In this paper we describe an approach to the problem. It generates appropriate names in Bulgarian from DBpedia URIs in other languages. These new names are used in two ways: (1) to form gazetteers for annotation of DBpedia instances in Bulgarian texts; and (2) they refer back to the DBpedia URIs from which they have been created and in this way provide access to all RDF statements about the original URIs.

The paper presents several transliteration models and their evaluation. Their evaluation is done over 100 examples, transliterated manually by two people, independently from each other. The discrepancies between the two human transliterations demonstrate the complexity of the task.

The structure of the paper is as follows: in section 2 we describe the problem in more detail; in section 3 we present some related approaches; section 4 reports on the preliminary experiments; section 5 describes how the training data is extended on the basis of the results from the preliminary models and new models are trained; section 6 provides some heuristic rules for combining transliteration and translation of names' parts; section 7 describes the evaluation of the models; the last section concludes the paper and presents some directions for future work.

2 Challenges

The transliteration of proper names presents many and quite challenging difficulties not only to automatic systems, but also to humans. A lot of information is needed to perform the task: the language of origin to determine the pronunciation; some real world facts like how this name was/is actually pronounced by the person it belongs or belonged to (in case of personal names) or by the locals (in case of toponyms) and so on; and also the tradition in transliterating names from this particular language into the given target language. Even if all of this information is gathered and if it is consistent (which is not always the case), there are still decisions to be made – the task of finding a phonetic equivalent of one language's phoneme into another is not trivial; in

[1] `http://wiki.dbpedia.org/`

some cases the name or parts of it are meaningful words in the source language and it might not be obvious which is more appropriate – translation or transliteration; and sometimes it might be better to leave the name in its original script.

In their survey on machine transliteration (Karimi et al., 2011) give a list of five main challenges for an automated approach to the task: *1*) script specifications, *2*) language of origin, *3*) missing sounds, *4*) deciding on whether or not to translate or transliterate a name (or part of it), and *5*) transliteration variants.

Fortunately, *1)* does not present great difficulties when dealing with European languages only, as the direction of writing is the same and the characters do not undergo changes in shape due to the phonetic environment. The only problem related to the script (apart from the minor one of choosing appropriate encoding) is the letter case. We decided to leave the upper and lower case characters in our training data, trading overcomplication of the statistical models for a result that does not need additional postprocessing.

On the other hand, *2)* is a very hard challenge and we decided to leave it aside for now. When we extract names from Wikipedia, we know the language they are written in, but there are no straightforward ways to figure out the language they came from.

3) is the challenge of finding a phonetic match for a sound that does not exist in the target language. Human translators need to make a decision which of the phonemes at hand is most appropriate in the particular case, and *appropriate* does not necessarily mean *phonetically close*, as orthography and etymology could also be considered important. We hope to overcome this difficulty by training machine translation Moses models on parallel lists of names where the decisions about sound mapping have already been made by humans.

Challenge *4)* is related to the fact that transferring a name from one language to another can happen not only through transliteration, but also via translation (for example, until 1986, the French name *Côte d'Ivoire* has been translated, not transliterated in Bulgarian as *Бряг на слоновата кост*) or direct adoption (like many music band names).[2] To distinguish the cases where transliteration is needed from those where direct adoption or translation is more appropriate, we use some heuristics that involve the combination of several different models and are described in more details in section 6.

Problem *5)* is very peculiar. It looks similar to the variation in translation, but is different in its 'abnormality'. One would accept as normal different translations of a single sentence and we know that even in the source language the meaning of this sentence can be expressed in other ways. Transliteration, on the other hand, is expected to produce one single result for each name, just as this name is an unvaried reference to an entity. However, there are often multiple transliterations of the same name. Our models do not attempt to generate all the acceptable variants, but we calculate how different our results are from manually generated transliterations and we expect this estimation to be useful to determine if an expression is likely to be a transliteration of a certain name.

3 Related Works

(Matthews, 2007) approaches transliteration very similarly to the way we do. Like us, he trains machine translation Moses models on parallel lists of proper names. The language pairs for which he obtains transliteration and backtransliteration models are English and Arabic, and English and Chinese. Unlike him, we are only interested in forward transliteration to Bulgarian. And our approach differs from his in several other aspects. First, we do not lowercase our training data. Second, we explore not only unigram models, but also bigram ones. And finally, we employ heuristics to decide whether to transliterate or not.

The construction of our models was inspired by (Nakov and Tiedemann, 2012). They train the only transliteration models for Bulgarian known to us. They use automatic transliteration as a substitution for machine transla-

[2]Or it might be a combination of the three. Here we will not deal with mixed cases as such. We will consider as cases of direct adoption only those where the whole name has been directly adopted. We will treat as translation cases all cases where at least some part of the name has been translated, and we will take the rest to be transliteration ones.

tion between very closely related languages, namely Bulgarian and Macedonian. Their models are of several different types – unigram, bigram, trigram – and their results show that bigrams perform best, because they are "a good compromise between generality and contextual specificity" (Nakov and Tiedemann, 2012, p. 302). We have also trained and compared unigram and bigram models (see Sections 4 and 5), however we left the trigrams out because of the specificity problem (a trigram generally occurs much less often than a unigram, for example), which gets even worse with proper names coming from many different languages with very diverse letter sequence patterns.[3]

Here we will not give a full overview of the automatic transliteration techniques, instead we will reference (Karimi et al., 2011), which is a detailed survey on the topic, and (Choi et al., 2011), where the main approaches to the task are explained and compared.

4 Preliminary Transliteration Models

We have trained several machine translation Moses[4] models. We have used standard settings for Moses baseline models for this purpose. The language models have been trained on the Bulgarian part of the English – Bulgarian parallel list of names. The translation models were obtained in two steps. The first step was to train models on the data we had, cleaned and tidied as much as possible (details are given in section 4.1).

The second step was to apply the first models on the data to further clean and tidy it up (details are given in section 5.1), so that a second, better series of models is obtained. In this section, we describe the first models, and the next section deals with the ones that were the product of the second step.

4.1 Training Data

The parallel lists of names on which we have trained our models have been extracted from DBpedia. We have used the instance type feature to select the URLs of all people, places and organizations in seven languages: Bulgarian, English, German, French, Russian, Italian and Spanish. Then we have mapped the Bulgarian names to the corresponding foreign names via the interlanguage links in DBpedia. We have further enlarged the lists by adding Airpedia[5] entries with assumed types 'Person', 'Place' or 'Organization' that were not present in DBpedia.

The obtained parallel lists have been cleaned from potential noise. Bulgarian entries that did not contain any Cyrillic letters were removed, as these are not cases of transliteration, but rather adoption of a foreign spelling. We used a Bulgarian word form dictionary (Popov et al., 2003) to detect and exclude probable translation cases.[6] We have also removed name pairs with mismatching number of words to avoid confusing the model if two names of a person are given in one language and only one in the other.

At the end, for each language paired with Bulgarian we had name lists with the following lengths:

English	38,360	German	30,899
Friench	30,446	Italian	27,369
Spanish	25,312	Russian	21,256

[3]We have trained several trigram models, but they performed poorly in our development tests, which was strong enough reason for us to drop them. The following table shows the BLEU scores that the different models obtained for each source language they were applied on. (The abbreviations representing the model names are clarified in section 5, here 'T' stands for 'trigram'.)

	en	**fr**	**de**	**it**	**ru**	**es**
PUM	86.31	85.63	86.84	86.36	90.01	86.25
PBM	81.94	81.45	83.30	82.02	86.37	82.25
PTM	73.79	73.79	76.94	74.56	81.22	74.39
UUM	88.63	88.56	88.77	87.76	87.24	87.80
UBM	83.76	85.02	85.77	84.47	83.30	84.46
UTM	76.76	77.52	78.77	77.80	78.71	77.01
BUM	88.29	88.12	88.67	88.11	87.65	88.07
BBM	84.50	85.32	85.54	84.73	84.04	84.52
BTM	76.78	77.52	78.77	77.80	77.71	77.01
TUM	88.17	88.40	88.79	87.86	87.31	87.94
TBM	84.42	84.72	85.44	84.62	83.34	84.48
TTM	77.41	77.69	78.82	77.70	78.97	77.09

[4]`http://www.statmt.org/moses/`

[5]`http://www.airpedia.org/`, this is an automatically generated extension of DBpedia.

[6]A minor problem here is that some foreign names look like a Bulgarian word when transliterated. We extracted the 100 most frequent meaningful word forms from the lists of Wikipedia articles we had and we filtered 20 of them that are more likely to have been obtained via transliteration, not translation. These 20 words were not treated as meaningful ones and the names containing them were not excluded from our lists.

4.2 Models Trained

The data was divided into training (80%), tuning and development sets (10% each). We have trained 12 preliminary transliteration models – two for each source language: one unigram model and one bigram model. The names in the training sets for the unigram models looked like this:

(*E l v i s*)

The training data for the bigram model looked like this:

(*E El lv vi is s*)

The opening bracket indicates the beginning of the word and the closing bracket indicates the end of the word.

5 Main Transliteration Models

5.1 Training Data

Before the training of the preliminary models, the data was cleaned from all name pairs with mismatching number of words. After obtaining the first transliteration models, we were able to put back these parts of the names that were present in both languages. We detected which word corresponds to which by transliterating (with the preliminary models) the foreign name to Bulgarian and comparing this transliteration to the original Bulgarian name. The words that were similar enough (were at NLD less than 0.1[7]) were considered as an indication that the source word and the Bulgarian word correspond to each other, and thus were retained. For example, the English name *A. J. Kronin* and the Bulgarian counterpart *Арчибалд Кронин* both contain the family name of the person, but in Bulgarian the first name is given in its full form, and in English it is abbreviated as well as the second and they would only confuse the models if they are present in the training data.

Another problem that we were able to solve with the help of the preliminary models were the swapped names (some languages prefer to put the given name before the family name, other not). We again calculated the similarity between each word in the transliteration and in the original Bulgarian name, to determine if rearrangement is needed and to perform it.

[7]Normalized Levenshtein Distance, see Section 7 for an explanation of the metric.

As we have two preliminary models – unigram and bigram – we obtained two new data sets – one enhanced by the unigram models, and one enhanced by the bigram models.

The application of the unigram models on the name lists lead to the following, larger data sets:

English	43,673	German	34,014
French	33,807	Italian	31,619
Spanish	29,579	Russian	27,980

The application of the bigram models also enlarged the initial data sets with similar success:

English	43,608	German	34,004
French	33,944	Italian	31,620
Spanish	29,520	Russian	27,994

5.2 Models Trained

On each of the new enhanced data sets we have trained 12 new models, altogether 24. The models that we used to improve the training sets will be called from now on preliminary unigram and preliminary bigram model (PUM and PBM). The unigram models trained on the data, amended with the help of the preliminary unigram models, will be called unigram enhanced unigram models or UUM for short. Similarly, we will have bigram enhanced unigram models (BUM), unigram enhanced bigram models (UBM), and bigram enhanced bigram models (BBM).

6 Ensemble Approach

What we train our models, for is *how* to transliterate. To decide *if* transliteration is needed, we do not employ statistical approach, but the following heuristics.

6.1 First Heuristic

We use this heuristic to resolve the *direct addoption vs. trasnliteration* problem. When one name is the same in all source languages, we assume that this name should stay as it is in Bulgarian too, and not be transliterated.[8] One example would be the band name *Skazi* that

[8]It is very important here that we have Russian, a language using Cyrillic script among our languages. Languages that use the same alphabet are more likely to directly adopt each other's proper names and if we only relied on Latin script, we would have concluded that direct addoption in Bulgarian is more appropriate in most cases, which is not desirable.

our heuristics leave in Latin script, because it is given like this in all of the source languages.

6.2 Second Heuristic

We use this heuristic to resolve the *translia-tion vs. trasnliteration* problem. If the results we got from all models are all different, then we assume that this name has been translated, not transliterated in our source languages and needs a translation in the target language too. An example from our evaluation set would be the *Romanian Television*, whose name is always transliterated differently by our models depending on the language it comes from.[9]

6.3 Voting

We assume that in the rest of the cases, trasnliteration is the appropriate method to transfer a name to Bulgarian. We apply voting to decide which transliteration (the one from which source language) to take. In case of a tie, a random choice is made.

7 Evaluation

We have evaluated the transliteration models on a small set of 100 names. The names were taken from the English DBpedia and we made sure that there are DBpedia entries for these entities in the other five languages too. We asked volunteers to transfer the names to Bulgarian by whatever method they find more appropriate: transliteration, translation or direct adoption of the foreign name. The 100 names were divided in portions of 10 and each portion was transliterated by two different volunteers.[10] In this way, we obtained two different references to compare the automatically generated transliterations to. From now on, we will refer to these two references as REF1 and REF2.

[9]With this heuristic we aim at detecting cases of at least partial translation and we do not attempt to identify which parts of the name are translated and which are not.

[10]This division of the data in portions is not relevant to our evaluation method, it is only a way to speed up the gathering of human input. Transliteration is one of the most time consuming subtasks of translation and together with the research it takes quite a lot time and effort, which we opted to bring to a minimum for our volunteers.

7.1 General Evaluation

The measure we use is normalized (by the length of the longer name) Levenshtein distance (NLD). We have chosen it because it is very intuitive – a 0-distance means that the two names are absolutely the same, 1 means that they are completely different and all the values in between can be interpreted as the proportion of errors. 0.1 for example means that there is one error (a letter that needs to be removed, inserted or substituted to obtain the correct name) for each ten letters. This measure is also fair to long names, unlike the simple Levenshtein distance.[11]

If our models have chosen wrongly whether transliteration is needed or not for a particular name, they get NLD score 1 for it. Further down in this section we present a separate evaluation of the heuristics that detect translation or direct adoption cases. So, for each name the distance between the automatic and the human generated transliteration is calculated as follows:

$$NLD = \begin{cases} \frac{LD(w_{aut}, w_{ref})}{|max(w_{aut}, w_{ref})|}, & \text{for correct decision to transliterate} \\ 0, & \text{for correct decision to translate/adopt} \\ 1, & \text{for wrong decision.} \end{cases}$$

where $LD(w_{aut}, w_{ref})$ is the well known Levenshtein distance between the words w_{aut} (which is the automatic generated transliteration) and w_{ref} (the human generated reference), i.e. the minimal number of edit operations (deletions, insertions and substitutions) that can transform w_{aut} into w_{ref}.

In our evaluation we present mean NLD for all the 100 names, the percentage of the names that received NLD score exactly 0, as well as the percentage of those that received score less than 0.1.

Table 1 shows how different from each other the two sets of manual transliterations are.

[11]The measures we use are very similar to the ones chosen by (Matthews, 2007, pp. 29-46) with the only difference that we normalize the Levenshtein distance. Other measures exist too, for example the ones recommended for the shared transliteration task in 2012 (Zhang et al., 2012, pp. 4-5).
We feel free to refine the measure we use and not stick to one that has been previously employed, because there are no other proper name transliteration systems for Bulgarian, to which we could compare our results anyway.

mean NLD	**zeroes**	**less than 0.1**
0.173	57%	68%

Table 1: Comparison of REF1 and REF2.

model	**mean NLD**	**zeroes**	**< 0.1**
PUM	0.256	39%	51%
PBM	0.234	42%	**57%**
UUM	0.242	41%	53%
UBM	0.270	36%	51%
BUM	0.286	37%	51%
BBM	**0.222**	**43%**	54%

Table 2: Comparison of the performance of the automatic models to REF1.

'Mean NLD' is the mean normalized Levenshtein distance for the 100 names, 'zeros' is the percentage of name pairs with NLD equal to zero, and 'less than 0.1' is the percentage of pairs for which NLD is less or equal to 0.1.

It is to note that only 57% of the name pairs in the two reference sets are absolutely the same (NLD=0). This is due to the 'abnormal' variation of transliterations that was mentioned as one peculiar challenge in Section 2.

We have compared the results of our automatic ensemble approach to each of the two references. Tables 2 and 3 present how different the machine transliteration is from respectively REF1 and REF2.

Generally, the automatic transliterations are not more different from the references than the references are from each other. From Table 2 it seems that BBM performs best, as it has lowest mean NLD and a greater percentage of exact matches. However, the models that are closest to the second reference are different. It is not clear if the enhanced models are altogether better or worse than the preliminary ones, but in most cases the best result is presented by one of the enhanced models.

model	**mean NLD**	**zeroes**	**< 0.1**
PUM	0.226	41%	56%
PBM	0.184	43%	57%
UUM	0.225	37%	55%
UBM	0.184	**46%**	59%
BUM	**0.180**	44%	**60%**
BBM	0.188	44%	58%

Table 3: Comparison of the performance of the automatic models to REF2.

model	F_{ref1}	F_{ref2}
PUM	0.64	0.67
PBM	0.56	0.50
UUM	0.79	0.67
UBM	0.71	0.68
BUM	0.69	0.64
BBM	0.55	0.50

Table 4: Evaluation of the 'translation vs. transliteration' heuristic

7.2 Evaluation of the Heuristics

The tables above present an overall evaluation of our approach. It might be interesting, though, to look into the performance of our heuristics separately.

The first heuristic detects cases where direct adoption is to be preferred over transliteration. It relies solely on the input in the source languages, so it produces the same results for all models.

Against the first reference we have calculated F score $F_{ref1} = 0$, and against the second reference we obtained $F_{ref2} = 0.75$. This result is more odd than bad, which can be explained quite well by the absent inter-annotator agreement for this task ($\kappa = -0.03$).[12]

The second heuristic resolves the 'translation or transliteration' problem and it is dependent on the output of the transliteration models, which is why we present F scores for each model separately in Table 4 (again there are two F scores, because there are two references). The inter-annotator agreement for this task is almost perfect, $\kappa = 0.89$.

It is worth noticing that BBM, which seemed to be performing best of all models, is the worst one in this task.

7.3 Evaluation of the Transliteration Models Alone

The names from the evaluation list which were considered by both human transliterators to

[12]There is not a single case in which the two human transliterators both think that a name should be left in its original script. This reveals that there is an ongoing process in Bulgarian to establish when direct adoption is more appropriate than transliteration.

mean NLD	**zeroes**	**less than 0.1**
0.107	57.7%	71.8%

Table 5: Comparison of the two manually generated transliterations for the 78 pure transliteration cases.

need pure transliteration, were 78. On them we have calculated mean NLD, percent of zero scores and percent of low scores (< 0.1), as in the general evaluation of our approach, to be able to present this time an evaluation of the transliteration models without the impact of the heuristics. Table 5 shows how close the two references are to each other for the pure transliteration cases only, Tables 6 and 7 present the transliterations of the automatic models compared to REF1 and REF2 respectively. There are six models of a kind, one for each of the six different source languages.

Generally, all models seem to perform better when they only need to transliterate, not to decide whether to transliterate. The gap between the references is also narrower. REF2 is again more similar to the automatic transliterations than REF1. It is interesting that REF2 is in average closer to the automatic models than to REF1 for English (all models) and German (almost all models).

All the models perform best when applied to English as source language. This might be due to the fact that the training data for English is more than for any of the other languages. Another reason for this result might be that the list of names, that was presented to the human transliterators, was extracted from the English DBpedia. Even though the volunteers were encouraged to use Wikipedia as resource also in the other five languages, they might have had somewhat of an inclination towards a more English sounding transliteration.

It is not very easy to explain why Russian as source language challenges the automatic models so much. One would expect that transliteration between two languages using the same alphabet would be easier, but it is exactly the opposite here. The two languages have very different pronunciation rules and even use quite different sets of phonemes, which is one possible reason why a name is transliterated with one sequence of letters in Russian and a different one in Bulgarian.

Now, when we have the evaluation of the different models for each source language, we can start working on a weighted version of the voting in our ensemble approach. It would be interesting to see, if giving more weight to the English and German models, and less to the Russian ones would contribute significantly to the final results.

8 Conclusion and Future Work

We have presented an approach to machine transliteration that makes use of machine translation Moses models and some simple heuristics to detect if transliteration is appropriate, and to perform it if it is. This approach can help closing the gap between well and not so well represented languages in DBpedia. Even though the transliterations generated by our models would be somewhat different than manual transliteration, one could still make use of them. For example, in an information retrieval task, one could search not for exact matches of the name, but for words that are very similar. (Our evaluation can serve as a guide to what similarity should be taken as being enough. It would be nice to have at some point a larger set of human generated references for a sounder result.) Besides, if integrated in a machine translation system, our transliteration approach would give a (close to) acceptable result, improving in this way the performance of the whole machine translation system.

One problem that we have not tackled yet is determining the language of origin for each name. When we do, we could train different models according to this information and see if automatic transliteration benefits from it as much as a human transliterator does.

Another improvement would be to train models for more languages to extend the coverage of our approach. It is also interesting to experiment with different groups of languages and see how the number and kind of the source languages influences the results of our ensemble approach. Experiments with weighted voting for our ensemble approach would also be beneficial.

model		**mean NLD**	**zeroes**	**< 0.1**
PUM	de	0.131	39.5%	56.6%
	en	**0.114**	**44.7%**	**57.9%**
	es	0.156	36.8%	44.7%
	fr	0.142	34.2%	55.3%
	it	0.162	32.9%	48.7%
	ru	0.412	22.4%	27.7%
PBM	de	0.132	36.8%	53.9
	en	**0.114**	**39.5%**	**59.2%**
	es	0.157	34.2%	48.7%
	fr	0.134	**39.5%**	56.6%
	it	0.167	30.2%	48.7%
	ru	0.407	26.3%	31.6%
UUM	de	**0.120**	42.1%	**57.9%**
	en	0.126	**43.4%**	56.6%
	es	0.150	36.8%	48.7%
	fr	0.139	39.5%	51.3%
	it	0.158	38.1%	50.0%
	ru	0.411	17.1%	22.4%
UBM	de	0.122	39.5%	56.6%
	en	**0.116**	**43.4%**	53.2%
	es	0.147	34.2%	48.7%
	fr	0.135	38.3%	**57.9%**
	it	0.156	34.2%	55.3%
	ru	0.403	23.7%	27.6%
BUM	de	0.128	40.8%	58.9%
	en	**0.123**	**43.4%**	**59.2%**
	es	0.158	34.2%	43.4%
	fr	0.132	43.4%	57.9%
	it	0.160	36.8%	47.4%
	ru	0.405	19.7%	27.6%
BBM	de	0.125	39.5%	**60.5%**
	en	**0.120**	**44.7%**	59.2%
	es	0.156	32.9%	59.2%
	fr	0.135	38.1%	55.3%
	it	0.172	31.6%	48.7%
	ru	0.392	26.3%	30.3%

Table 6: Comparison of the performance of the automatic models to REF1 for the pure transliteration cases.

model		**mean NLD**	**zeroes**	**< 0.1**
PUM	de	0.103	38.2%	57.9%
	en	**0.096**	**42.1%**	**60.5%**
	es	0.124	38.2%	51.3%
	fr	0.117	35.5%	51.3%
	it	0.141	31.6%	50.0%
	ru	0.400	21.1%	26.3%
PBM	de	0.109	34.2%	53.9
	en	**0.098**	**36.8%**	**57.9%**
	es	0.134	32.9%	53.9%
	fr	0.119	**36.8%**	53.9%
	it	0.142	27.6%	50.0%
	ru	0.402	23.7%	26.3%
UUM	de	**0.097**	40.8%	59.2%
	en	0.103	**42.1%**	**60.5%**
	es	0.121	39.5%	52.6%
	fr	0.109	39.5%	53.9%
	it	0.140	34.2%	48.7%
	ru	0.398	18.4%	21.1%
UBM	de	0.102	36.8%	59.2%
	en	**0.098**	**42.1%**	**61.8%**
	es	0.121	35.5%	50.0%
	fr	0.117	35.5%	56.6%
	it	0.133	32.9%	56.3%
	ru	0.394	25%	27.6%
BUM	de	0.108	36.8%	57.9%
	en	**0.103**	**40.8%**	**59.2%**
	es	0.127	35.5%	48.7%
	fr	0.108	34.4%	55.3%
	it	0.139	34.2%	48.7%
	ru	0.392	22.4%	25.0%
BBM	de	**0.101**	36.8%	**60.5%**
	en	0.105	**40.8%**	57.9%
	es	0.129	32.9%	47.4%
	fr	0.117	36.8%	55.6%
	it	0.154	28.9%	44.7%
	ru	0.379	26.3%	30.3%

Table 7: Comparison of the performance of the automatic models to REF2 for the pure transliteration cases.

Acknowledgments

This research has received partial support by the EC's FP7 (FP7/2007-2013) project under grant agreement number 610516: "QTLeap: Quality Translation by Deep Language Engineering Approaches" and FP7 grant 316087 AComIn "Advanced Computing for Innovation", funded by the European Commission in 2012–2016.

We are grateful to all the volunteers for their input, also to Petya Ossenova and Hristo Todorov for revising an early draft of this paper.

We thank the three anonymous reviewers, whose remarks, comments, suggestions and encouragement helped us to improve the initial version of the paper. All errors remain our own responsibility.

References

Key-Sun Choi, Hitoshi Isahara, and Jong-Hoon Oh. 2011. A comparison of different machine transliteration models. *CoRR*, abs/1110.1391.

Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. 2011. Machine transliteration survey. *ACM Comput. Surv.*, 43(3):17:1–17:46, April.

David Matthews. 2007. Machine transliteration of proper names. Master's thesis, School of Informatics, University of Edinburgh.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 301–305, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dimityr Popov, Kiril Simov, Svetlomira Vidinska, and Petya Osenova. 2003. *Spelling Dictionary of Bulgarian*. Nauka i izkustvo, Sofia, Bulgaria.

Min Zhang, Haizhou Li, Ming Liu, and A Kumaran. 2012. Whitepaper of news 2012 shared task on machine transliteration. In *Proceedings of the 4th Named Entity Workshop*, NEWS '12, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.