An Unsupervised Method for Discovering Lexical Variations in Roman Urdu Informal Text

Abdul Rafae, ‡Abdul Qayyum, ‡Muhammad Moeenuddin,
Asim Karim, †Hassan Sajjad and ‡Faisal Kamiran,
†Qatar Computing Research Institute, Hamad Bin Khalifa University
Lahore University of Management Sciences, ‡Information Technology University

Abstract

We present an unsupervised method to find lexical variations in Roman Urdu informal text. Our method includes a phonetic algorithm *UrduPhone*, a feature-based similarity function, and a clustering algorithm *Lex-C*. UrduPhone encodes roman Urdu strings to their phonetic equivalent representations. This produces an initial grouping of different spelling variations of a word. The similarity function incorporates word features and their context. Lex-C is a variant of k-medoids clustering algorithm that group lexical variations. It incorporates a similarity threshold to balance the number of clusters and their maximum similarity. We test our system on two datasets of SMS and blogs and show an f-measure gain of up to 12% from baseline systems.

1 Introduction

Urdu is the national language of Pakistan and one of the official languages of India. It is written in Perso-Arabic script. However in social media and short text messages (SMS), a large proportion of Urdu speakers use roman script (i.e., the English alphabet) for writing, called *Roman Urdu*.

Roman Urdu lacks standard lexicon and usually many spelling variations exist for a given word, e.g., the word *zindagi* [life] is also written as *zindagee*, *zindagy*, *zaindagee* and *zndagi*. Specifically, the following normalization issues arise: (1) differently spelled words (see example above), (2) identically spelled words that are lexically different (e.g., *bahar* can be used for both [outside] and [spring], and (3) spellings that match words in English (e.g, *had* [limit] for the English word 'had'). These inconsistencies cause a problem of data sparsity in basic natural language processing tasks such as Urdu word segmentation (Durrani and Hussain, 2010), part of speech tagging (Sajjad and Schmid, 2009), spell checking (Naseem and Hussain, 2007), machine translation (Durrani et al., 2010), etc.

In this paper, we propose an unsupervised feature-based method that tackles above mentioned challenges in discovering lexical variations in Roman Urdu. We exploit phonetic and string similarity based features and incorporate contextual features via top-k previous and next words' features. For phonetic information, we develop an encoding scheme for Roman Urdu, *UrduPhone*, motivated from Soundex. Compared to other available phonetic-based schemes that are mostly limited to English sounds only, UrduPhone maps Roman Urdu homophones effectively. Unlike previous work on short text normalization (see Section 2), we do not have information about standard word forms in the dataset. The problem becomes more challenging as every word in the corpus is a candidate of every other word. We present a variant of the k-medoids clustering algorithm that forms clusters in which every word has at least a specified minimum similarity with the cluster's centroidal word. We conduct experiments on two Roman Urdu datasets: an SMS dataset and a blog dataset and evaluate performance using a gold standard. Our method shows an f-measure gain of up to 12% compared to baseline methods. The dataset and code are made available to the research community.

2 Previous Work

Normalization of short text messages and tweets has been in focus (Sproat et al., 2001; Wei et al., 2011; Clark and Araki, 2011; Roy et al., 2013; Chrupala, 2014; Kaufmann and Kalita, 2010; Sidarenka et al., 2013; Ling et al., 2013; Desai and Narvekar, 2015; Pinto et al., 2012). However, most of the work is limited to English or to other

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 823–828,
Lisbon, Portugal, 17-21 September 2015.

resource-rich languages. In this paper, we focus on Roman Urdu, an under-resourced language, that does not have any gold standard corpus with standard word forms. Therefore, we are restricted to the task of finding lexical variations in informal text. This is a rather more challenging problem since in this case every word is a possible variation of every other word in the corpus.

Researchers have used phonetic, string, and contextual knowledge to find lexical variations in informal text.[1] Pinto et al. (2012; Han et al. (2012; Zhang et al. (2015) used phonetic-based methods to find lexical variations. Han et al. (2012) also used word similarity and word context to enhance performance. Wang and Ng (2013) used normalization operations e.g., missing word recovery and punctuation correction to improve normalization process. Irvine et al. (2012) used manually prepared training data to build an automatic normalization system. Contractor et al. (2010) used string edit distance to find candidate lexical variations. Yang and Eisenstein (2013) used an unsupervised approach with log linear model and sequential Monte Carlo approximation.

We propose an unsupervised method to find lexical variations. It uses string edit distance like Contractor et al. (2010), Sound-based encoding like Pinto et al. (2012) and context like Han et al. (2012) combined in a discriminative framework. However, in contrast, it does not use any corpus of standard word forms to find lexical variations.

3 Our Method

The lexical variations of a lexical entry usually have high phonetic, string-based, and contextual similarity. We integrate a phonetic-based encoding scheme, UrduPhone, a feature-based similarity function, and a clustering algorithm, Lex-C.

3.1 UrduPhone

Several sound-based encoding schemes for words have been proposed in literature such as *Soundex* (Knuth, 1973; Hall and Dowling, 1980), *NYSIIS* (Taft, 1970), *Metaphone* (Philips, 1990), Caverphone (Wang, 2009) and Double Metaphone.[2] These schemes encode words based on their sound which in turn serves as grouping words of similar sounds (lexical variations) to one code. However, most of the schemes are designed for English and European languages and are limited when apply to other family of languages like Urdu.

In this work, we propose a phonetic encoding scheme, *UrduPhone*, tailored for Roman Urdu. The scheme is derived from the Soundex algorithm. It groups consonants on the basis of common homophones in Urdu and English. It is different from Soundex in two particular ways:[3] Firstly, UrduPhone generates encoding of length six compared to length four in Soundex. This enables UrduPhone to avoid mapping different forms of a root word to same code. For example, *muskurana* [smiling] and *mshuraht* [smile] encode to one form *MSKR* in Soundex but in UrduPhone, they have different encoding which are *MSKRN*, *MSKRHT* respectively. Secondly, we introduce consonant groups which are mapped differently in Soundex. We do this by analyzing Urdu alphabets that map to a single roman form e.g. words *samar* [reward], *sabar* [patience] and *saib* [apple], all start with different Urdu alphabets that have identical roman representation: *s*. In UrduPhone, we map all such cases to a single form.[4]

3.2 Similarity Function

The similarity between two words w_i and w_j is computed by the following similarity function:

$$S(w_i, w_j) = \frac{\sum_{f=1}^{F} \alpha^{(f)} \times \sigma_{ij}^{(f)}}{\sum_{f=1}^{F} \alpha^{(f)}}$$

Here, $\alpha^{(f)} > 0$ is the weight given to feature f, $\sigma_{ij}^{(f)} \in [0, 1]$ is the similarity contribution made by feature f, and F is the total number of features. In the absence of additional information, and as used in the experiments in this work, all weights can be taken equal to one. The similarity function returns a value in the interval [0, 1] with larger values signifying higher similarity.

We use two types of features in our method: word features and contextual features. Word features can be based on phonetics and/or string similarity. The phonetic similarity between words w_i and w_j is 1 (i.e., $\sigma_{ij} = 1$) if both words have the same UrduPhone ID or encoding; otherwise, their

[1] Spell correction is also considered as a variant of text normalization (Damerau, 1964; Tahira, 2004; Fossati and Di Eugenio, 2007). Here, we limit ourselves to the previous work on short text normalization.

[2] http://en.wikipedia.org/wiki/Metaphone

[3] Due to limited space, we limit the description of UrduPhone to its comparison with Soundex.

[4] A complete table of UrduPhone mappings is provided in the supplementary material.

similarity is zero. The string similarity between words w_i and w_j is defined as follows:

$$\sigma_{ij} = \frac{lcs(w_i, w_j)}{\min[len(w_i), len(w_j)] \times edist(w_i, w_j)}$$

Here, $lcs(w_i, w_j)$ is the length of the longest common subsequence in words w_i and w_j and $len(w_i)$ is the length of word w_i. $edist(w_i, w_j)$ returns the edit distance between words except when the edit distance is 0, in which case it returns 1.

Contextual features include top-k frequently occurring previous and next words' features. Let $a_1^i, a_2^i, \ldots, a_5^i$ and $a_1^j, a_2^j, \ldots, a_5^j$ be the word IDs for the top-5 frequently occurring words preceding word w_i and w_j, respectively. Then, the similarity between words is given by (Hassan et al., 2009)

$$\sigma_{ij} = \frac{\sum_{k=1}^{5} \rho_k}{\sum_{k=1}^{5} k}$$

Here, ρ_k is zero if a_k^i does not have a match in a_*^j (i.e., in the context of word w_j); otherwise, $\rho_k = 5 - \max[k, l] - 1$ where $a_k^i = a_l^j$ and l is the highest rank (smallest integer) at which a previous match had not occurred. Instead of word IDs in a_i's, UrduPhone IDs or string similarity based cluster IDs can be used to reduce sparsity and improve matches among similar words.

3.3 Lex-C: Clustering Algorithm

We develop a new clustering algorithm, called Lex-C, for discovering lexical variations in informal text. This algorithm is a modified version of the k-medoids algorithm (Han, 2005). It incorporates an assignment similarity threshold, $t > 0$, for controlling the number of clusters and their similarity. In particular, it ensures that all words in a cluster have a similarity greater than or equal to this threshold. It is important to note that the poular k-means algorithm is known to be effective for numeric datasets only which is not true in our case, and it cannot utilize our specialized similarity function for lexical variation discovery.

Specifically, Lex-C starts from an initial clustering based on UrduPhone or string similarity. It finds the centroidal word, w_c^k, for cluster k as the word with which the sum of similarities of all other words in the cluster is a maximum. Then, each non-centroidal word is assigned to the cluster k if $S(w_i, w_c^k)$ is a maximum among all clusters *and* $S(w_i, W_c^k) \geq t$. If the latter condition is not satisfied (i.e., $S(w_i, w_c^k) < t$) then instead of assigning word w_i to cluster k, it starts a new cluster. These two steps are repeated until convergence.

4 Experimental Evaluation

We empirically evaluate UrduPhone and our complete method involving Lex-C separately on two real-world datasets. Performance is reported with B-Cubed precision, recall, and f-measure (Bagga and Baldwin, 1998; Hassan et al., 2015) on a gold standard dataset. These performance measures are based on element-wise comparisons between predicted and actual clusters that are then aggregated over all elements in the clustering. This avoided the issue of 100% precision with low recall (all words belong to separate clusters) and 100% recall with low precision (all words belong to one cluster).

4.1 Dataset and Gold Standard

The first dataset, Web dataset, is scraped from Roman Urdu websites on news[5], poetry[6], SMS[7] and blog[8]. The second dataset, SMS dataset, is obtained from chopaal, an internet based group SMS service[9]. For evaluation, we use a manually annotated database of Roman Urdu variations (Khan and Karim, 2012). Table 1 shows statistics of the datasets in comparison with the gold standard.

Dataset	Web	SMS
Unique words	22,044	28,908
Overlap with Gold Standard	12,600	13,087
UrduPhone IDs	3,952	3,599

Table 1: Datasets and gold standard statistics. Overlap with gold standard = number of words appearing in gold standard; UrduPhone IDs = number of distinct UrduPhone encodings.

4.2 UrduPhone Evaluation

We compare UrduPhone with Soundex and its variants.[10] These algorithms are used to group words based on their encoding and then evaluated against the gold standard. Table 2 shows

[5] http://www.shashca.com,http://stepforwardpak.com/

[6] https://hadi763.wordpress.com/

[7] http://www.replysms.com/

[8] http://roman.urdu.co/

[9] http://chopaal.org

[10] We use NLTK-Trainer's phonetic library `http://bit.ly/1OJGL9Q`

the results on Web dataset. UrduPhone outperforms Soundex, Caverphone, and Metaphone while Nysiis's f-measure is comparable to that of UrduPhone. We observe that Nysiis produces a large number of single word clusters (out of 6,943 clusters produced 5,159 have only one word). This gives high precision but recall is low. UrduPhone produces fewer clusters (and fewer one word clusters) with high recall.

Algorithm	Pre	Rec	Fme
Soundex	0.30	0.97	0.46
Metaphone	0.49	0.80	0.64
Caverphone	0.31	0.92	0.46
Nysiis	0.63	0.69	0.66
UrduPhone	0.51	0.94	**0.67**

Table 2: Comparison of UrduPhone with other algorithms on Web dataset

4.3 Lex-C Evaluation

We compared Lex-C with k-means and EM clustering algorithms. With both of these algorithms we used the same feature set (i.e., word features, phonetic features, and contextual features), However, their performance lagged the performance of our approach. The primary reason for this is that our feature space is not continuous while k-means and EM algorithms work best for continuous feature spaces.

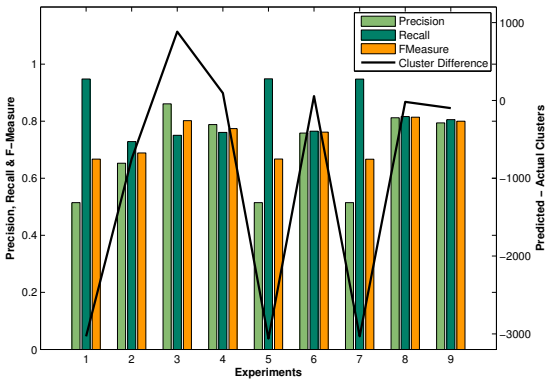

Figure 1: Performance results for Web dataset

4.4 Performance of our Method

We conduct extensive experiments to evaluate the performance of our method. We vary initial clusterings (UrduPhone encoding or string similarity based clustering); evaluate various combinations of phonetic, string, and contextual features; and consider different previous/next top-5 words' features (word ID, cluster ID, and/or UrduPhone ID). Table 3 gives details of each experiment setting.

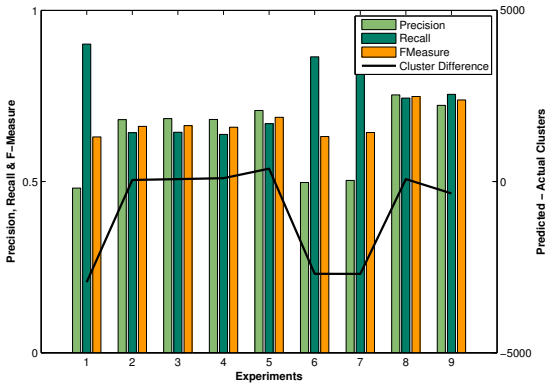

Figure 2: Performance results for SMS dataset

ID	Initial	Word	Context-1	Context-2
1	–	UPhone	–	–
2	–	String	–	–
3	String	String	Word ID	–
4	String	String	Cluster ID	–
5	String	UPhone	Cluster ID	–
6	UPhone	UPhone	Cluster ID	–
7	UPhone	UPhone	Word ID	–
8	UPhone	UPhone	UPhone ID	Word ID
9	UPhone	UPhone	UPhone ID	Cluster ID

Table 3: Details of experiments' settings where Initial is initial clustering based on String or UrduPhone (UPhone)

Figures 1 and 2 show results of selected experiments for Web and SMS datasets respectively. The x-axis shows the experiment (Exp.) IDs while the left y-axis gives the precision, recall, and f-measure and the right y-axis shows the difference between the number of predicted and actual clusters. Exp. 1 and 2 are baselines corresponding to UrduPhone encoding (UPhone ID) and string similarity based word clustering (Cluster ID) respectively. The remaining experiments have different initial clustering, word features, and up to two contextual features. In these results, the similarity threshold t is selected such that the number of discovered clusters is as close as possible to the number of actual clusters in the gold standard for each dataset. This is done to make the results comparable across different settings.

Compared to baselines, our method shows a gain of up to 12% and 8% in Web and SMS datasets respectively. The best performances are obtained when UrduPhone is used as a feature and UrduPhone IDs are used to define the context (Exp. 8 and 9). In particular, when both Urdu-

Phone IDs and word IDs/cluster IDs are used for contextual information (i.e, with two sets of top-5 previous and next words' features) the f-measure is consistently high.

We analyzed the performance of Exp. 8 (best settings for Web dataset) with varying t and showed it in Figure 3. It is observed that the value of t controls the number of clusters smoothly, and precision increases with the number of clusters and f-measure reaches a peak when number of clusters is close to that in the gold standard.

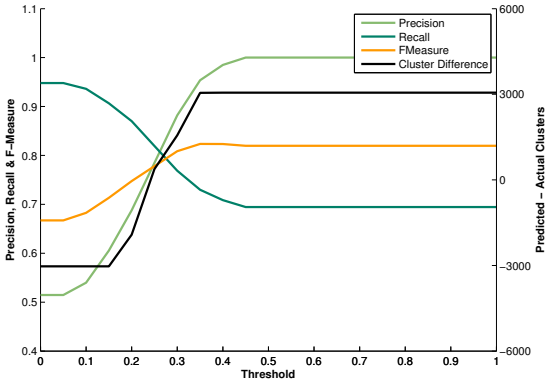

Figure 3: Effect of varying threshold t on Web dataset (experiment 8)

5 Conclusion

We proposed an unsupervised method for finding lexical variations in Roman Urdu. We presented a phonetic encoding scheme UrduPhone for Roman Urdu, and developed a feature-based clustering algorithm Lex-C. Our experiments are evaluated on a manually developed gold standard. The results confirmed that our method outperforms baseline methods. We made the datasets and algorithm code available to the research community.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Grzegorz Chrupala. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 680–686.

Eleanor Clark and Kenji Araki. 2011. Text normalization in social media: Progress, problems and applications for a pre-processing system of casual english. *Procedia-Social and Behavioral Sciences*, 27:2–11.

Danish Contractor, Tanveer A Faruquie, and L Venkata Subramaniam. 2010. Unsupervised cleansing of noisy text. In *Proceedings of the 23rd International Conference on ComputationalLinguistics: Posters*, pages 189–196. Association for Computational Linguistics.

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3), March.

Neelmay Desai and Meera Narvekar. 2015. Normalization of noisy text data. *Procedia Computer Science*, 45:127–132.

Nadir Durrani and Sarmad Hussain. 2010. Urdu word segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 528–536, Los Angeles, California, June. Association for Computational Linguistics.

Nadir Durrani, Hassan Sajjad, Alexander Fraser, and Helmut Schmid. 2010. Hindi-to-urdu machine translation through transliteration. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 465–474, Uppsala, Sweden, July. Association for Computational Linguistics.

Davide Fossati and Barbara Di Eugenio. 2007. A mixed trigrams approach for context sensitive spell checking. In *Computational Linguistics and Intelligent Text Processing*, pages 623–633. Springer.

Patrick A. V. Hall and Geoff R. Dowling. 1980. Approximate string matching. *ACM Comput. Surv.*, 12(4):381–402.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432. Association for Computational Linguistics.

Jiawei Han. 2005. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Malik Tahir Hassan, Khurum Nazir Junejo, and Asim Karim. 2009. Learning and predicting key web navigation patterns using bayesian models. In *Computational Science and Its Applications–ICCSA*, pages 877–887. Springer.

Malik Tahir Hassan, Asim Karim, Jeong-Bae Kim, and Moongu Jeon. 2015. Cdim: Document clustering by discrimination information maximization. *Information Sciences*.

Ann Irvine, Jonathan Weese, and Chris Callison-Burch. 2012. Processing informal, romanized pakistani text messages. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12, pages 75–78. Association for Computational Linguistics.

Max Kaufmann and Jugal Kalita. 2010. Syntactic normalization of twitter messages. In *International Conference on Natural Language Processing, Kharagpur, India*.

Osama Khan and Asim Karim. 2012. A rule-based model for normalization of sms text. In *IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece, November 7-9, 2012*, pages 634–641.

Donald E Knuth. 1973. *The Art of Computer Programming: Volume 3, Sorting and Searching*. Addison-Wesley.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. In *EMNLP*, pages 73–84.

Tahira Naseem and Sarmad Hussain. 2007. A novel approach for ranking spelling error corrections for urdu. *Language Resources and Evaluation*, 41(2):117–128.

Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language Magazine*, 7(12):39–44, December.

David Pinto, Darnes Vilariño Ayala, Yuridiana Alemán, Helena Gómez-Adorno, Nahun Loya, and Héctor Jiménez-Salazar. 2012. The soundex phonetic algorithm revisited for SMS text representation. In *Text, Speech and Dialogue - 15th International Conference, TSD 2012, Brno, Czech Republic, September 3-7, 2012. Proceedings*, pages 47–55.

Sudipta Roy, Sourish Dhar, Saprativa Bhattacharjee, and Anirban Das. 2013. A lexicon based algorithm for noisy text normalization as pre processing for sentiment analysis. *International Journal of Research in Engineering and Technology*, 02.

Hassan Sajjad and Helmut Schmid. 2009. Tagging urdu text with parts of speech: A tagger comparison. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, pages 692–700, Athens, Greece.

Uladzimir Sidarenka, Tatjana Scheffler, and Manfred Stede. 2013. Rule-based normalization of german twitter messages. In *Proc. of the GSCL Workshop Verarbeitung und Annotation von Sprachdaten aus Genres internetbasierter Kommunikation*.

Richard Sproat, Alan W. Black, Stanley F. Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

R. Taft. 1970. Name search techniques. *Special report. Bureau of Systems Development,New York State Identification and Intelligence System*.

Naseem Tahira. 2004. A hybrid approach for Urdu spell checking.

Pidong Wang and Hwee Tou Ng. 2013. A beam-search decoder for normalization of social media text with application to machine translation. In *HLT-NAACL*, pages 471–481.

John Wang, editor. 2009. *Encyclopedia of Data Warehousing and Mining, Second Edition (4 Volumes)*. IGI Global.

Zhongyu Wei, Lanjun Zhou, Binyang Li, Kam-Fai Wong, Wei Gao, and Kam-Fai Wong. 2011. Exploring tweets normalization and query time sensitivity for twitter search. In *TREC*.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 61–72.

Xin Zhang, Jiaying Song, Yu He, and Guohong Fu. 2015. Normalization of homophonic words in chinese microblogs. In *Intelligent Computation in Big Data Era*, pages 177–187. Springer.