Procesamiento del Lenguaje Natural, Revista nº 52 marzo de 2014, pp 45-52 recibido 15-11-13 revisado 06-02-14 aceptado 07-02-14

Lexical Normalization of Spanish Tweets with Rule-Based Components and Language Models

Normalización léxica de tweets en español con componentes basados en reglas y modelos de lenguaje

Pablo Ruiz, Montse Cuadros and **Thierry Etchegoyhen**
Vicomtech-IK4
Mikeletegi Pasealekua 57
Parque Tecnológico de Gipuzkoa, 20009 Donostia/San Sebastián
{pruiz,mcuadros,tetchegoyhen}@vicomtech.org

Abstract: This paper presents a system to normalize Spanish tweets, which uses preprocessing rules, a domain-appropriate edit-distance model, and language models to select correction candidates based on context. The system is an improvement on the tool we submitted to the Tweet-Norm 2013 shared task, and results on the task's test-corpus are above-average. Additionally, we provide a study of the impact for tweet normalization of the different components of the system: rule-based, edit-distance based and statistical.
Keywords: Spanish microtext, lexical normalization, Twitter, edit distance, language model

Resumen: Este artículo presenta un sistema para la normalización de tweets en español, que usa reglas de preproceso, un modelo de distancias de edición adecuado al dominio y modelos de lenguaje para seleccionar candidatos de corrección según el contexto. Se trata de un sistema mejorado basado en el que presentamos en la tarea compartida Tweet-Norm 2013. El sistema obtiene resultados superiores a la media en el corpus de test de la tarea. Presentamos además un estudio del impacto en la normalización de los diferentes componentes del sistema: basados en reglas, en distancia de edición, y estadísticos.
Palabras clave: microtexto, español, castellano, normalización léxica, Twitter, distancia de edición, modelo de lenguaje

1 Introduction

Studies on the lexical normalization of Spanish microtext are scarce, e.g. Armenta et al. 2003, which predates Twitter and focuses on SMS. Newer studies are (Pinto et al., 2012) and (Oliva et al., 2013), which also focus on SMS. Other recent studies are (Mosquera et al., 2012), which discusses the normalization of Spanish user-generated context in general, and (Gómez Hidalgo et al., 2013), which presents a detailed microtext tokenization method that can be employed for normalization.

A larger body of literature exists for English microtext normalization (see Eisenstein, 2013 for a review). Some approaches rely on large amounts of labelled training data, e.g. (Beaufort et al., 2010) and (Kaufmann and Kalita, 2010), which examine SMS normalization. However, such resources are not available for Spanish. An approach that performs normalization of English Tweets without the need of annotated data is Han and Baldwin, 2011.

As an initiative to explore the application of different microtext normalization approaches, and to help overcome the lack of resources and tools for such a task in Spanish, SEPLN 2013 hosted the Tweet-Norm Workshop[1] (Alegría et al. 2013a).

The system for Spanish tweet normalization presented in this study comprises data resources to model the domain, as well as analysis modules. It is an improvement on the tool we submitted (Ruiz et al., 2013) to the Tweet-Norm 2013 shared task.

The paper is organized as follows: the system's architecture and components are presented in Section 2, resources employed in Section 3, and settings and results-evaluation in Section 4. Conclusions and future work are discussed in Section 5.

[1] http://komunitatea.elhuyar.org/tweet-norm/

ISSN 1135-5948

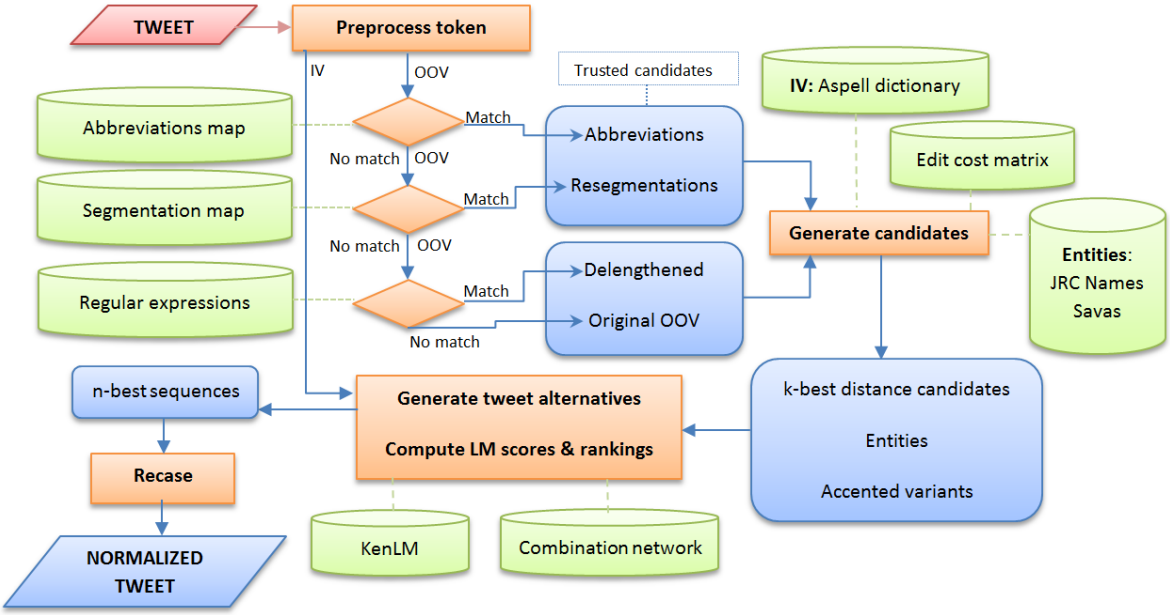

Figure 1: System Architecture

2 *Architecture and components*

The system's architecture and components are shown in Figure 1 and explained in following.

2.1 Rule-Based preprocessing

The preprocessing module was rule-based, relying on 110 hand-crafted mappings between patterns that match out-of-vocabulary (OOV) items and a correction for the expressions matched by the patterns. The mappings were implemented as case-insensitive regular expressions.

The first set of mappings (46 rules) was used to identify abbreviations, and expand them if needed. A second set was used to resegment tokens commonly written together in microtext (21 rules).

The final set of mappings (43 rules) detected emoticons and delengthened OOV items with repeated characters, besides mapping OOVs to DRAE[2] onomatopoeias. Repeated letters were reduced to a single letter, unless a word with a repeated letter was found in Aspell's Spanish inflected form dictionary (v1.11.3)[3]. E.g. *vinoo* was preprocessed to *vino*, but *creeeen* was reduced to *creen*.

These regex-based mappings were based on the most common errors in a corpus of 1 million tweets crawled by ourselves and spellchecked with Hunspell (v1.3.2). Microtext expressions such as RT (retweet) or HT (hat tip) were considered in-vocabulary.

2.2 Correction-candidate generation

The correction candidates generated were validated against a dictionary for in-vocabulary (IV) items, and against entity lists.

2.2.1 Dictionary candidates

The base-form ($Base_{ED}$) to generate candidates from was either the original OOV or the preprocessed form of the OOV.

Prior to candidate generation, $Base_{ED}$ was lowercased if all of its characters were in uppercase and it had a length of more than three characters.

Candidates were generated for $Base_{ED}$ using two methods: minimum edit distance and regular expressions. With both methods, the candidates that were not found in Aspell's dictionary were rejected and did not proceed to further steps in the normalization workflow.

[2] Spanish Academy dictionary, www.rae.es
[3] *aspell -l es dump master | aspell -l es expand*

Using minimum-edit distance (Damerau, 1964), up to two case-insensitive character edits (insertions, deletions or substitutions) were performed on the edit-base form $Base_{ED}$. The cost of each edit operation was not uniform: edits that result in correcting a common error were given a lesser cost than edits that correct uncommon errors. This method is context-insensitive: the cost of an edit operation did not take into account the characters adjacent to those undergoing the edit, or the position in the word of the characters being edited (word-initial, word-final, etc.).

However, context sensitivity is useful in candidate generation and candidate scoring, since the frequency of certain errors depends on context; e.g., *d*-deletion is more frequent in participle endings *-ado*, *-ido* than elsewhere. To add context-sensitivity at character level to the model, we generated candidates via regexes that repair common errors. A custom distance-scoring scheme was created for these regex-based candidates.

If both the edit-distance and the regex-based method returned the same candidate, and distance scores differed, the score chosen for the candidate was the smaller one.

2.2.2 Entity Candidates

For each OOV, a caps-initial variant and a variant with all characters in uppercase were generated, and looked up in entity lists. The OOV itself was also looked up. Matches were stored as entity candidates.

2.3 Candidate selection

The goal of candidate selection is to choose a single correction for each OOV, among the set containing the candidates created in the pre-processing and candidate-generation steps, as well as the original form of the OOV itself.

The original OOV is one of the forms to consider: It is part of the normalization workflow to decide whether to keep the unmodified OOV as the normalized form, or to propose an edited variant.

The output of the candidate selection method is a single candidate, C_{Nopos}, which stands for *final candidate pending postprocessing*.

The terminology used in the description of the algorithm (below) is the following:

- *Trusted Candidates*: candidates from the *Abbreviations* or *Resegmentation* mappings in the preprocessing step.
- *Untrusted Candidates*: candidates obtained with the methods in *a* through *c* below.
  a. *DelenCand:* obtained in preprocessing with *Delengthening* rules.
  b. *DistCands*: candidates, along with their distance to their $Base_{ED}$ form, generated with either context-sensitive or context-insensitive character-edits (see section 2.2.1)
  c. *EntCand*: a candidate from entity-detection heuristics (section 2.2.2).
- *LMCands*: When more than one untrusted candidate exists for an OOV, *LMCands* is the subset of the OOV's candidates which is ultimately assessed against the language model, in order to choose an optimal candidate for the OOV.
- *Accented Variant*: for this algorithm, a string S_1 is an accented variant of a string S_2 if they match in a case-and-accent-insensitive manner: *mía* is an accented variant of *Mia*, as is *mañana* of *Manana*.

In essence, the algorithm first selects a subset of the correction candidates for each OOV in the tweet. Then, if more than one candidate exists for some OOV in the tweet, a language model (LM) scores candidate combinations at tweet level, assessing best fit. The algorithm is presented below, and explanations and examples follow it.

The operations in A through C below take place **for each OOV** in the tweet.

A. Initial Filtering

1. Filter the *DistCands* set in two steps:

   1.1. Candidates at a distance higher than 1.5 (configurable threshold) from their $Base_{ED}$ are filtered out.

   1.2. Among the remaining candidates in *DistCands*, all of the candidates at the smallest distance present in the set are retained. E.g. if candidates at distance 0.5 and 0.8 exist, candidates at distance 0.8 are filtered out.

B. Trusted Candidates

2. If a correction candidate was obtained in preprocessing, via *Abbreviation* mappings, (see Section 2.1), it is selected as C_{Nopos} (the final candidate pending postprocessing).
3. If a correction candidate was obtained in preprocessing via *Resegmentation* mappings, it is selected as C_{Nopos}.

C. Untrusted Candidates

4. If a correction candidate of type *EntCand* exists, add it to the *LMCands* set.

    4.1. If among the candidates in *DistCands*, *accented variants* exist for an *EntCand* candidate, add them to *LMCands*.

5. If a correction candidate was obtained in preprocessing, via *Delengthening* regexes, and the candidate is IV, add it to the *LMCands* set.

    5.1. If among the candidates in *EditCands*, *accented variants* exist for the *Delenghtening* candidate, add them to the *LMCands* set.

6. If no candidate has been selected so far (i.e. no trusted candidates exist, and the *LMCands* set is empty), add the content of the *DistCands* set (already filtered in step 1) to *LMCands*.

7. If *LMCands* is empty, select the original OOV form as C_{Nopos}.

After steps 1 to 7 have applied for each OOV in the tweet, candidates are assessed at tweet level.

D. Tweet-Level Scoring

Once each OOV in the tweet has been resolved into a trusted candidate, an *LMCands* set, or the original OOV form as a default, the following procedure applies, **at tweet level**.

1. If each OOV in the tweet has one candidate only, that candidate is chosen and moves to postprocessing.
2. Otherwise, with each combination of candidates from the different OOVs' *LMCands* sets, tweet alternatives are created and scored against the language model. Candidates, the combination of which maximizes log probability for the whole tweet-alternative containing them, are chosen, and move to post-processing.

In the initial filtering stage, step 1.1 eliminates candidates whose edit-distance from their $Base_{ED}$ is too high for them to be likely corrections. Step 1.2 is similar in the sense that it narrows down the candidates to another k-best subset in terms of distance. Accuracy on both development and test-sets improved significantly with both steps included in the workflow.

Trusted candidates result from matches against mappings and rules created by a human domain-expert, for unambiguous cases. They can thus be reliably promoted to C_{Nopos} status.

Unlike the previous case, *untrusted candidates* represent ambiguous cases, and forms that have been generated through automatic means. Better accuracy is obtained when statistical methods and string comparison metrics are employed to assess their validity.

Entity-candidates (*EntCand*) are added to the *LMCands* set when available. Additionally, since accent omission is a very frequent error, we also consider accented variants of *EntCand*. E.g. for *EntCand Rio*, accented variant *río* is considered.

IV candidates output by *Delengthening* regexes may also require disambiguation. For instance, the correct variant of the form *si*, obtained from delengthening OOV *siii*, could be *si*, or *sí*, depending on context. Thus, accented variants for such IV items are added to *LMCands*.

For *EntCand* and *Delenghtening* candidates, it is the language model's task to decide between accented or unaccented variants.

For *DistCand* candidates, the language model disambiguates among the k-best candidates in terms of distance score.

2.4 Postprocessing (Recasing)

Once the above processes have applied, the case of the candidate selected may still be incorrect; this can happen when the case of the original OOV was incorrect, and was not corrected earlier in the workflow (e.g. a tweet-initial OOV starting with lowercase). A candidate may have also undergone decasing via regex application or candidate-set generation, which were deployed in a case-insensitive manner.

For these reasons, a postprocessing was performed, whereby the selected candidate was uppercased if one of the following four conditions applied.

1. If it was in tweet-initial position.
2. If it was the second token in the tweet, and the first token was a mention (*@user*) or hashtag (*#topic*).
3. If the previous token was a sentence delimiter[4].
4. In all other positions, the first character of the selected candidate was uppercased if the original OOV's first character was in uppercase.

3 Resources

In-vocabulary (IV) items were determined using the Aspell dictionary (v1.11.3).

Entity lists were obtained from the JRC Names[5] database. A list of named entities manually annotated in the Spanish subset of the SAVAS[6] corpus (Del Pozo et al., to appear) was also used. The Spanish subset of SAVAS consists of 200 hours of Spanish news broadcasts from 2012. It contains entities from current events, often discussed on Twitter.

Normalization does not require entity classification or linking, but merely identifying whether a token belongs to an entity or not. Accordingly, in our entity lists multiword entities were split into their tokens. Tokens for which a lowercase variant exists in Aspell's dictionary were filtered out.

For measuring candidate distance, a cost matrix for character edits was created. Additionally, a custom distance-scoring scheme was devised for candidates obtained with regular expressions at the candidate-generation stage (see Section 2.2.1).

For the edit-cost matrix, costs were domain-specific, estimated by surveying the frequency of character substitutions in Spanish tweets. For instance, editing *k* as *q* (as in one of the editing steps needed to correct frequent error *kiero* as *quiero*) was assigned a lesser cost than uncommon edits. Costs were also inspired by (Ramírez and López, 2006), who found that 51.5% of spelling errors in Spanish were accent omissions. Accordingly, a cost model was created where replacing a non-accented character with its accented variant cost less than other substitutions. Table 1 provides example costs. Using the table, editing *alli* to *allí* costs 0.5; *kiero* to *quiero* costs 1.5.

Error	Correction	Cost (each)
a, e, i, o, u, n	á, é, í, ó, ú, ñ	0.5
k, *null*	q, u	0.75
p, a, z	m, u, k	1

Table 1: Edit Costs

Besides the edit-cost matrix, a set of regular expressions was created, to model context-sensitive corrections (for errors that are very frequent in specific contexts only, like *d*-dropping in participles), and for corrections involving one-to-many character edits. A custom scoring scheme was created to assess distance for these corrections.

The goal of the custom scoring was for regex-based corrections to receive smaller costs than edit-distance would assign to them. For instance, consider correcting *parxe* as *parche*. Using regexes, this was modeled as a single *x→ch* one-to-many character edit, with a cost of 0.5, rather than two one-to-one character edits *x→c* and *ø→h*, which would lead to a higher correction cost.

Thus, editing *parxe* into *parche* (which repairs a very common error in the domain), costs 0.5, less than editing *parxe* into a less likely correction like *parte*, with a cost of 1. In the way just described, the custom scoring scheme was designed to favour corrections that are likely in the domain.

Table 2 shows some of the corrections modeled via regexes, and their costs. Note that corrections for some *spelling-pronunciations* (i.e. correcting *p* as *pe*, or *k* as *ca*) were also modeled with regexes.

Error	Correction	Cost (each)
ki, x, wa, ni	qui, ch, gua, ñ	0.5
ao$	ado	0.5
p, t, k	pe, te, ca	0.5

Table 2: Context-Sensitive and One-to-Many character Edit Costs

In terms of language models, we created a 5-gram case-sensitive language model with Kenlm[7] (Heafield, 2011), using an *unk* token. The model was based on the OpenSubs Spanish corpus, available at the Opus repository (Tiedmann, 2009), pruned to 31 million subtitles, merged with 1 million tweets containing IV tokens only, collected by

[4] The delimiters considered were **. ! ? " …**

[5] optima.jrc.it/data/entities.gzip

[6] www.fp7-savas.eu/savas_project

[7] kheafield.com/code/kenlm/

ourselves according to the procedure described below.

The tweets in the corpus were prepared as follows: tweets with language value *es* and European time zones were collected in the spring of 2013. Only tweets for which Hunspell (v1.3.2) detected no errors were accepted. In order to decrease false positives, Hunspell dictionaries were enriched with entity lists. Tweet tokenization largely treated emoticons, URLs and repeated punctuation as single tokens. For tweets where there was at least 70% of token-overlap with other tweets, only one exemplar was accepted.

The choice to use subtitles was motivated by our experiments for the Tweet-Norm workshop, which showed that results for language models trained on subtitles showed similar accuracy to the results for language models trained with tweets containing IV-only items.

4 Results and evaluation

Accuracy was 71.15% on the Tweet-Norm shared-task test-corpus (564 tweets and 662 annotated OOVs)[8]. For reference, average accuracy on the task, based on the scores obtained by the 13 participating systems, was 56.16%, the range being 33.5% to 78.1%.

The improvement that each module achieves over the baseline is provided in Table 3, in terms of accuracy and increase in percentage points (ptp). The baseline (19.78%) is the score attained when accepting all OOV forms as correct.

The results on Table 3 support the conclusion that both the rule-based preprocessing and the edit-distance based candidate generation were useful. Applied in isolation, rule-based preprocessing achieved gains of 17.8 ptp over the baseline, and edit-distance in isolation obtained an improvement of 14.81 ptp.

The results also support the conclusion that the candidate filtering procedure and the language model managed to disambiguate among candidates successfully, achieving gains of 44.11 ptp over the baseline (without post-processing; gains after post-processing are 51.37 ptp).

As regards the edit-distance component, a relevant result is that, using a cost-model that reflects common errors in the domain and integrates context-sensitive edits obtains better results than using a cost model where all edit costs are uniform. For instance, as Table 3 shows, the edit-distance module alone[9], with costs adapted to the domain, achieves an improvement of 13.6 ptp over the baseline. The gain increases to 14.81 ptp over the baseline if context-sensitive corrections are added.

Modules	ACCU (%)	GAINS (*ptp*)
Baseline	**19.78**	
Rule-Based Preprocessing Only	**37.61**	**+17.83**
Abbreviations + Resegmentations	26.28	
Abbreviations + Resegmentations + Delenghtening	37.61	
Edit Distance Only	**34.59**	**+14.81**
Generic Levenshtein	29.45	+9.67
Domain-Adapted Levenshtein (Context Insensitive)	33.38	+13.6
Domain-Adapted Levenshtein + Context-Sensitive Distance	34.59	+14.81
Entities Only	**21.45**	**+1.67**
All + Language Model		
No Postprocessing (recasing)	63.89	+44.11
With Postprocessing (recasing)	**71.15**	**+51.37**

Table 3: Normalization accuracy for each module in isolation and after LM application

However, if we use a generic distance model where all edits have a cost of 1, improvement over the baseline is 9.67 ptp only: 5.14 ptp below the cost-model adapted to the domain. The finding that context-sensitive corrections improve accuracy agrees with results by (Hulden and Francom, 2013).

Regarding candidate selection, one of the difficulties in applying language models in order to correct microtext is the abundance of other OOVs in the context of the OOV undergoing normalization in each case. Our previous normalization system compared each OOV's local context with the language model. In cases where other OOVs were part of a given

[8] http://komunitatea.elhuyar.org/tweet-norm/files/2013/11/tweet-norm_es.zip

[9] Regarding the "edit-distance only" results in Table 3, several candidates may exist at the same distance. Distance being the only factor in these results, a random choice between candidates at the same distance was avoided by ranking candidates with their distance score weighted at 90% and their language-model unigram logprob weighted at 10%, for all distance models.

OOV's local context, the candidates' scores were limited to unigram probabilities and backoff, which could decrease accuracy.

The current language model implementation, which considers all possible candidate combinations in order to compute the best fit against the LM of the tweet's entire word sequence, is more successful at normalizing cases of adjacent OOVs than an LM workflow based on local context, as the following example illustrates.

Consider a tweet (from the test-set) containing the sequence *nainonainonahh me atozigah con* ***tuh comentarioh*** *los besoohh virtualeh*. Local-context lookup in the LM corrects the bolded phrase as *tu comentarios* (withouth number agreement). This is expectable, since *p(*tu*)* > *p(*tus*)* in the model (−2.66 vs. −3.45), and *p(*comentarios*)* > *p(*comentario*)*, (−5.77 vs. −6.02). Since OOVs *tuh* and *comentarioh* are surrounded by other OOVs, a local-context lookup will not benefit from contextual information, and will be restricted to a unigram probability.

By contrast, the current LM workflow, which considers all possible candidate-combinations and assesses the complete tweet against the LM, successfully normalizes the sequence as *tus comentarios*, since it is able to find the higher probability for the sequence respecting agreement: −6.47, vs. −9.19 for the sequence with broken agreement. The LM also disambiguated successfully accented variants, such as *si* vs. *sí*.

Another salient result is that, the simple postprocessing module, which deploys four recasing rules to capitalize the final candidate depending on sentence position and on the original OOV's case, yields an improvement of 7.26 ptp compared to results without postprocessing. This agrees with findings by (Alegría et al., 2013b), whose recasing rules are a subset of ours, and who report notable gains from applying recasing rules.

Regarding the small gain that occurs when activating the entity heuristics, note that about half the entity-OOVs in the corpus are already correct in the baseline. For the remaining entities, precision was acceptable: 75% in both sets. However, recall was weak: 41% in the development set and 52% in the test-set. For these reasons, entity detection yielded a smaller gain over the baseline than other modules.

Finally, the system's upper bound (proportion of correct candidates generated, even if they were not selected as final) was 84.54%, similar to the upper bound of 85.47% reported by (Ageno et al. 2013) for the same corpus. Some of the OOVs for which no correct proposal was generated were entities. In some other cases, preprocessing rules that would map the original OOV to a viable candidate were missing.

5 Conclusions and future work

We presented a system for the normalization of Spanish tweets. The system uses rules to expand abbreviations, resegment tokens and delengthen OOVs into forms closer to IV tokens. Candidates are generated based on weighted edit-distance. The edit-cost model was adapted to the domain: costs were estimated taking into account common errors in tweets. Besides context-insensitive edits, distance-scoring had some context-sensitive rules, reflecting the likelihood that an edit would lead to a correction in a given context. The domain-adapted cost-model was shown to be more accurate than a generic-domain unweighted edit-distance model. Candidates were also proposed based on entity lists.

To disambiguate between candidates, the entire word sequence of tweet-alternatives containing all possible correction-candidate combinations (among k-best candidates) was checked for best fit against a language model. This global, tweet-level LM lookup method was more successful at normalizing sequences of adjacent OOVs than a lookup method that exploits an OOV's local context only.

Regarding future work, the current resegmentation rules were hand-crafted and a statistical workflow (e.g. Alegría et al., 2013b) would be an improvement. Also, our entity-detection heuristics should be improved for recall. In terms of candidate selection, we used the language model to disambiguate candidates at the smallest distance available in the candidate set, as better accuracy was obtained that way. Extending the scope of LM disambiguation beyond k-best candidates, while also improving accuracy, is a topic for future research. Finally, only a small proportion of the current LM training corpus consisted of tweets. It would be relevant to verify if results improve with an LM trained on a large in-vocabulary corpus of tweets, with the language model reflecting domain-specific textual characteristics more closely.

6 References

Ageno, A., P. R. Comas, L. Padró, J. Turmo. 2013. The TALP-UPC approach to Tweet-Norm 2013. *Proceedings of the Tweet Normalization Workshop at SEPLN 2013*.

Alegría, I., N. Aranberri, V. Fresno, P. Gamallo, L. Padró, I. San Vicente, J. Turmo, and A. Zubiaga. 2013a. Introducción a la tarea compartida Tweet-Norm 2013: Normalización léxica de tuits en español. *Proceedings of the Tweet Normalization Workshop at SEPLN 2013*.

Alegría, I., I. Etxeberria, and G. Labaka. 2013b. Una cascada de transductores simples para normalizar tweets. *Proceedings of the Tweet Normalization Workshop at SEPLN 2013*.

Armenta, A., G. Escalada, J.M. Garrido, and M. A. Rodríguez. 2003. Desarrollo de un corrector ortográfico para aplicaciones de conversión texto-voz. *Procesamiento del Lenguaje Natural*, 31:65-72.

Beaufort, R., S. Roekhaut, L. A. Cougnon, and C. Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing SMS messages. *48th Annual Meeting of the Association for Computational Linguistics*, 770-779, Uppsala, Sweden

Damerau, F. 1964. A technique for computer correction of spelling errors. *Communications of the ACM*, 7(3): 171-176.

Del Pozo, A, C. Aliprandi, A. Álvarez, C. Mendes, J. P. Neto, S. Paulo, N. Piccinini, M. Rafaelli. To appear. SAVAS: Collecting, Annotating and Sharing Audiovisual Language Resources for Automatic Subtitling. To appear in *Proceedings of LREC 2014*.

Eisenstein, Jacob. 2013. What to do about bad language on the internet. *Proceedings of NAACL-HLT*, pp. 359-369.

Han, B. and T. Baldwin. 2011. Lexical normalisation of short text messages: makn sens a #twitter. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1: 368-378, Association for Computational Linguistics, Stroudsburg, PA, USA.

Heafield, K. 2011. KenLM: Faster and Smaller Language Model Queries. *Proceedings of the Sixth Workshop on Statistical Machine Translation*, 187-197. Edinburgh, Scotland, UK.

Hulden, M. and J. Francom. 2013. Weighted and unweighted transducers for tweet normalization. *Proceedings of the Tweet Normalization Workshop at SEPLN 2013*.

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

Gomez Hidalgo, J. M., A. A. Caurcel Díaz, and Y. Iñiguez del Rio. 2013. Un método de análisis de lenguaje tipo SMS para el castellano. *Linguamática*, 5(1):31-39

Mosquera, A., E. Lloret and and P. Moreda. 2012. Towards facilitating the accessibility of web 2.0 texts through text normalization. In *Proceedings of the LREC Workshop: Natural Language Processing for Improvign Textual Accessibility (NLP4ITA)*, pp 9-14, Istanbul, Turkey.

Oliva, J., J. I. Serrano, M. D. Del Castillo, and A. Iglesias. 2013. A SMS normalization system integrating multiple grammatical resources. *Natural Language Engineering*, 19:121-141, 1.

Pinto, D., D. Vilariño Ayala, Y. Alemán, Helena, N Loya, and H Jiménez-Salazar. 2012. The Soundex phonetic algorithm revisited for SMS text representation. In P. Sojka, A. Horak, I. Kopecek, and Karel Pala (eds.). *Text, Speech and Dialogue*, LNCS Vol. 7499:47-55. Springer.

Ramírez, F. and E. López. 2006. Spelling Error Patterns in Spanish for Word Processing Applications. *Proceedings of LREC 2006*, 93-98.

Ruiz, P., M. Cuadros and T. Etchegoyhen. 2013 Lexical Normalization of Spanish Tweets with Preprocessing Rules, Domain-Specific Edit Distances, and Language Models. *Proceedings of the Tweet Normalization Workshop at SEPLN 2013*.

Tiedmann, J. 2009. News from OPUS. A Collection of Multilingual Parallel Corpora with Tools and Interfaces. In N. Nicolov and K. Bontcheva (eds.) *Recent Advances in Natural Language Processing*, Vol. V:237-248. John Benjamins, Amsterdam.