Natural Language Engineering **1** (1): 1–26. Printed in the United Kingdom

Selection of Correction Candidates for the Normalization of Spanish User Generated Content

M. MELERO[a1], M. R. COSTA-JUSSÀ[a2] , P. LAMBERT[a1], M. QUIXAL[a3]

[a1] *Grup de Lingüística Computacional, Universitat Pompeu Fabra*
Roc Boronat, 138, 08018 Barcelona, Catalunya, Spain
{*maite.melero,patrik.lambert*}*@upf.edu*
(Work partially done while at Barcelona Media, Spain)
[a2] *Institute for Infocomm Research, Human Language Technology Group*
1 Fusionopolis Way, 21-01 Connexis (South Tower), 138632 Singapore
vismrc@i2r.a-star.edu.sg
[a3] *Department of Spanish and Portuguese, The University of Texas at Austin*
150 W 21st Street, Austin, TX, 78712, United States of America
marti.quixal@utexas.edu

(*Received XXX; revised XXX*)

Abstract

We present research aiming to build tools for the normalization of User-Generated Content (UGC). We argue that processing this type of text requires the revisiting of the initial steps of Natural Language Processing (NLP), since UGC (micro-blog, blog, and, generally, Web 2.0 user generated texts) presents a number of non-standard communicative and linguistic characteristics – often closer to oral and colloquial language than to edited text. We present a corpus of UGC text in Spanish from three different sources: Twitter, consumer reviews and blogs, and describe its main characteristics. We motivate the need for UGC text normalization by analyzing the problems found when processing this type of text through a conventional language processing pipeline, particularly in the tasks of lemmatization and morphosyntactic tagging.

Our aim with this paper is to seize the power of already existing spell and grammar correction engines and endow them with automatic normalization capabilities, in order to pave the way for the application of standard NLP tools to typical UGC text. Particularly, we propose a strategy for automatically normalizing UGC by adding a module on top of a pre-existing spell checker that selects the most plausible correction from an unranked list of candidates provided by the spell checker. To build this selector module we train four language models, each one containing a different type of linguistic information in a trade off with its generalization capabilities. Our experiments show that the models trained on truecase and lowercase word forms are more discriminative than the others at selecting the best candidate. We have also experimented with a parametrized combination of the models, both by optimizing directly on the selection task and by doing a linear interpolation of the models. The resulting parametrized combinations obtain results close to the best performing model but do not improve on those results, as measured on the test set. The precision of the selector module in ranking number one the expected correction proposal on the test corpora reaches 82.5% for Twitter text (baseline 57%) and 88% for non-Twitter text (baseline 64%).

1 Introduction

The Web 2.0 has become a channel where users exchange, explain or write about their lives and interests, give opinions and comment on other people's opinions, most of the time using a casual language. This language often presents peculiarities that make it much closer to transcriptions of oral language than to standard edited text. Opinion mining techniques, just to mention an example, are becoming an important source of information for market research, social network analysis or learning analytics (Pang and Lee, 2008). In order to mine data or extract information from Web 2.0 we first need to understand its contents. Shortened or misspelled words, which are frequent in the Social Media informal style, increase the ambiguity and interpretation possibilities for the same word form, and pose a challenge to the achievement of certain Natural Language Processing (NLP) analysis tasks such as tokenization, part-of-speech (PoS) tagging or Named Entity Recognition, as reflected in the literature – (Foster et al., 2011; Muñoz-García and Navarro, 2012; Maynard et al., 2012; Aminian et al., 2012).

NLP techniques, which are used to provide linguistic representations from unstructured data, are typically developed to deal with standard language and may not yield the expected results on User Generated Content (UGC) text. As we discuss later in this section, several approaches have been tried to tackle this problem either by adapting the tools to the text or else by adapting the text to the existing tools, through a process generally known as 'text normalization'.

The structure of the article is as follows. We start by explaining our motivation and reviewing related work. We then present our findings on what characterizes UGC text in Spanish, based on a corpus study. We then explore the problems caused by UGC text to the performance of a pre-existing NLP tool for the analysis of Spanish text, by comparing the results of parsing two versions of the same UGC text: as-is and manually corrected. Finally, we present and evaluate an approach to text normalization that uses a module for the automatic selection of correction candidates; this module is built on top of a pre-existing spell checker.

1.1 Motivation

Pang and Lee (2008) show the relevance that the analysis of UGC text has for the analysis of product and service reputation and political trends. While the processing of large amounts of UGC data on the basis of frequencies of (mainly) words or correlated structured data (e.g., the use of star-based recommendation scales) have successfully characterized global opinion and reputation trends, depending on the granularity of the task at hand a more fine-grained characterization is required – e.g., Pang and Lee (2008), Gianfortoni et al. (2011). The non-standard characteristics of UGC text seem to be one of the main obstacles and challenges in this respect (Ritter et al., 2011; Maynard et al., 2012; Muñoz-García and Navarro, 2012).

It is often said that UGC text, as found in social media sites, resembles more oral language than regular edited text, such as news – which is the kind of text on and for which most NLP tools have been trained. However, studies of computer-mediated communication in the field of descriptive linguistics suggest that social media text is a language mode in itself (Herring, 2012a). The linguistic characterization of UGC text is a research line that might provide interesting insights into the automatic analysis of UGC – see Bender et al. (2011), Gouws et al. (2011) or Eisenstein (2013).

Researchers in opinion mining and sentiment analysis often attribute the inaccuracies of their NLP-based systems to the lower performance of the standard NLP tools when applied to noisy,

or ill-formed, language – though Bermingham and Smeaton (2010) argue that text length and noise might be less of a disadvantage when performing sentiment analysis on Twitter texts. Several studies have investigated the effect of UGC characteristics on the performance of standard NLP tools. Some have found that UGC text characteristics affect the performance of tasks such as topic detection (Muñoz-García and Navarro, 2012), constituency parsing and dependency parsing (Foster, 2010; Foster et al., 2011), Named-Entity Recognition (NER) (Ritter et al., 2011; Muñoz-García and Navarro, 2012; Maynard et al., 2012), and tokenization (Aminian et al., 2012). Different text cleansing or normalization strategies have been pursued to improve the performance of standard NLP tools for their application to social media text.

The contributions of this article are i) the presentation and characterization of a UGC corpus in Spanish, and ii) the description of a normalization strategy that, exploiting the power of pre-existing spell and grammar correction tools, allows for broad-coverage normalization including the normalization of the tokenization of non-standard words. In this respect, this paper supposes a novelty both in the strategies for normalization and in the inclusion of tokenization in the normalization task.

1.2 Related work

In this section, we review research on three different topics: the definition of text normalization as a task, the different approaches to text normalization and the strategies found in the literature for ranking correction candidates in an automatic correction task, as opposed to a human-machine interactive correction task.

1.2.1 Text normalization as a task

Text normalization is a term that has been used in the field to refer to a task that consists of transforming an original piece of text into a different piece of text in which some words have been converted to a *normal* form. The definition of norm, or normal form, depends on the task to be performed after normalization. Thus, Sproat et al. (2001), in one of the classical papers on the topic, present work on the normalization of text to be given as input to a text-to-speech synthesis engine, and claim that a similar approach could be applied to speech recognition and information extraction tasks.

However, the norm of a text-to-speech synthesis is not necessarily the same as the norm for information extraction – to the extent that misspellings are not tackled in Sproat et al. (2001)'s article, for speech synthesis requires a word form to be obtained that can be 'said aloud', while information extraction requires a word form that can be interpreted or (lexically) analyzed. This poses the interesting question of how linguistic 'norm' is actually defined, a topic that we briefly take up below.[1] A recent paper by Eisenstein (2013) addresses the issue of how the different ways in which social media text deviates can be influenced by different aspects: keyboard used, writer's literacy, writer's intentions –pragmatics–, medium's length, and actual social variables. This paper interestingly suggests how UGC, and again Twitter in particular, can be seen as a real-time observation platform of language as an evolving social communication tool.

[1] Arguably one might not be able to obtain a normal form for text-to-speech without the normal *written* form, but the task's goals are certainly different.

Another interesting issue is the type of text on which the normalization task is performed. While Sproat et al. (2001) work with four different types of text (newspaper text, real estate ads, and servlist texts on the topics of palmtop computers and cooking recipes), most later papers deal either with SMS texts – see for instance Choudhury et al. (2007), Kobus et al. (2008), or Cook and Stevenson (2009) – or Twitter text – see for instance Clark and Araki (2011), Brody and Diakopoulos (2011), Foster et al. (2011), Han and Baldwin (2011), Hassan and Menezes (2013) or Eisenstein (2013). Also, Liu et al. (2012) have worked on both SMS and Twitter datasets.

1.2.2 Approaches to text normalization

We find two main different ways of dealing with UGC text processing, which can be complementary. The first one focuses on adapting tools and algorithms to this type of text by annotating UGC corpora and training the corresponding tools from scratch (Michelson and Knoblock, 2005; Ritter et al., 2011), while the second one focuses on transforming the source UGC text into a normalized version before handling it over to the processing tools (Kobus et al., 2008; Agarwal et al., 2011; Liu et al., 2012). There are some attempts at combining both methods by adapting or extending the training models while, at the same time preparing the input text, by normalizing some of the UGC related phenomena (Foster, 2010; Foster et al., 2011).

Kobus et al. (2008) present an interesting discussion on three different "metaphors" or ways of looking at SMS language, a type of text that has some features in common with UGC text. Each of these views motivates a different approach to the normalization task. In the first approach, each input token is taken as a deviation of the correct word form, and normalization is thus viewed as a spell checking task. The second metaphor considers SMS language as a different language, and so normalization can be viewed as a machine translation task. Finally, it is possible to consider normalization as a speech recognition task because some people consider SMS as being closer to oral productions than to regular written texts. In fact, SMS spellings tend to be a closer approximation to the phonemic representation of a word than to its normative spelling, though in this context "typography and orthography take over the functions of sound" (Herring, 2012b).

Most of the attempts at text normalization are based on empirically obtained lists of frequently observed phenomena, as in Foster (2010), Kobus et al. (2008) and Agarwal et al. (2011), though some research applies more generalist approaches (Alonso, 2010). Zhu et al. (2007) present an approach to text normalization (of e-mails and blog posts) that uses a Conditional Random Fields algorithm with a lot of features and compare it with two other classical normalization methods. Zhu et al. (2007)'s approach tackles the normalization task as a tagging problem, where the different normalization transformations required are performed on each token depending on its type (line break, space, punctuation, word and special). Clark (2003) and Clark and Araki (2011) use rule-based approaches while Henríquez Q. and Hernández (2009) approach the task using a statistical machine translation system trained on original texts and their semi-automatically corrected version. More recently, Hassan and Menezes (2013) have shown how a version of Markov Random Walks can be used to train a normalizer on a small manually corrected corpus – without any further labeling.

Finally, Liu et al. (2012) propose what they call a broad-coverage normalization algorithm, meaning it should be domain independent, which includes a module grounded on visual priming theories that favor the occurrences of words closer or related to words that have already been seen in the text. They combine this strategy with letter sequence modeling –that to a certain extent

resonates as a simplification of Zhu et al. (2007)'s approach– and standard spell checking techniques to generate lists of alternatives to non-standard tokens, though they assume normalized tokenization.

Our aim in the present work is to seize the power of already existing spell and grammar correction engines and endow them with automatic normalization capabilities, in order to pave the way for the application of standard NLP tools to typical UGC text. In this respect, our research is closer to Liu et al. (2012)'s research, since they aim at the implementation of a broad coverage normalization strategy.

1.2.3 Correction candidate selection

N-gram models have been used for the detection[2] and correction[3] of misspellings in isolated words since the late 1980's (Kukich, 1992). Church and Gale (1990) demonstrate the potential of word bigrams to improve the accuracy of isolated word correction Mays et al. (1991) used trigram models and obtained 76% accuracy in detection and 73% accuracy in correction. Hodge and Austin (2003) integrate Hamming distance and n-gram algorithms that have high recall for typing errors and a phonetic spell-checking algorithm in a single architecture. Ahmed et al. (2010) propose a spell checker that works by selecting the most promising candidates from a ranked list that is derived from n-gram statistics and lexical resources. Other approaches that correct spelling include rule-based techniques (Mangu and Brill, 1997), a noisy channel model (Brill and Moore, 2000; Toutanova and Moore, 2002) and a ternary tree search (Martins and Silva, 2004). As far as we know, little work has been made to date on the subject for Spanish, with the exception of Alonso (2010). Alonso uses a normalization approach based on a strategy to group words into clusters by computing the Levenshtein edit distance between each new (unknown) word and a series of word clusters – her work is rather a classification task for out-of-vocabulary words, more than a candidate selection task.

2 Corpus-based characterization of UGC text in Spanish

As a reference corpus to study UGC related phenomena, we have collected a sample of texts in Spanish from the following sources: blogs (collected using Google Blog Search), hotel reservations (booking.com), consumer reviews in three different domains (ciao.es) and Twitter. In the following sections, we describe the corpus size, source distribution and the annotation rationale.

2.1 Corpus size and distribution

The total size of the corpus is 7,583 sentences, or 192,417 words. Table 1 shows the size of the corpus in terms of sentences and words, and the corresponding percentage of each source. The Ciao site comprises texts from three different domains: cars (61%), mobile operators (12%) and banking (26%). The last row shows the ratio between the number of running words (total number of words) and the vocabulary size (number of different words appearing at least once, i.e. word types). The figures indicate that Twitter and Booking texts present a greater lexical variation –

[2] Detection is the task of finding words or expressions that do not appear in a specific lexicon or that do not respond to a particular morphological pattern for word formation.
[3] Correction is the task of providing one or more proposals for a given error, possibly as a ranked list.

which can be accounted for by these being less topic-focused types of texts compared to ciao.es texts.

	Total	Twitter	Blog	Booking	Ciao
No. of sentences	7583	20%	21%	2%	57%
No. of words	192417	14%	23%	1%	62%
Ratio words/word types	6	3.1	6	3.4	7.6

Table 1. *Corpus size in sentences and words, and ratio of word types (the lower the ration, the higher is the number of word types per total number of words).*

2.2 Annotation rationale

The normalization of ill-formed text presupposes a definition of norm. The concept of norm may vary from one linguistic community to the other. For example, norm is reached by consensus in the English-speaking world, but it is dictated by a prescriptive institution for Spanish (Real Academia Española de la Lengua, RAE) or French (Académie Française). In addition, within a particular community or corporation it can be further restricted with arbitrary norms, as is the case of *controlled languages* for machine-translated manuals or *corporate guidelines* in publishing houses. In the English speaking world of NLP, one could argue that the norm for tagging and parsing is the Wall Street Journal, since it has become the *de facto* standard.

In our case the norm is the one underlying our dictionary, which mostly follows the RAE, and thus includes the most common Latin American spellings and words. For instance, our dictionary includes both *futbol* (strength on the *o*) and *fútbol* (strength on the *ú*). From the perspective of NLP, the norm is further restricted by the arbitrary decisions regarding the linguistic apparatus. If the processing pipeline is not able to recognize one particular word it will not identify it as belonging to the language; and it will fail to assign the corresponding lexical and morphosyntactic features.

As suggested in the introduction, the normalization of text for text-to-speech (TTS) is also different from the normalization of text for text mining. For instance, in a TTS task the highway name *m40*, as written in the tweet captured in Figure 1, would eventually result into *eme cuarenta* [em forty], while in an text mining text it would stop in *M-40* – which might be an intermediate state for the TTS task.

We agree with Eisenstein (2013) in that we argue that "[n]ormalization is often impossible without changing the meaning of the text" given the social and pragmatic implications that UGC text entails.

2.3 Annotation procedure

The reference corpus was manually corrected. Textual deviations were reviewed, corrected and classified by three different annotators (none of which is involved in the research work). Each

Fig. 1. Real twitter text containing the highway name *m40* for *M-40*.

annotator annotated a different set of texts, so inter-annotator agreement cannot be calculated. Texts were not randomly assigned to annotators, but all annotators ended up annotating at least two different text genres or domains. Each deviation from standard language norms, was identified, classified and corrected. They were instructed to perform their task in two steps:

Step 1 Localization and correction
1. Search for deviating forms
2. Mark the span of the deviating form using square brackets
3. Write the alternative normalized version of the text within the brackets
4. Write the number of tokens originally involved in the deviation

Step 2 Classification
5. Classify the deviating form according to the following typologies:
   (a) Linguistic type
   (b) Transformation type

Annotators, all of whom had a degree in either theoretical or applied linguistics, were given the instructions in print with no examples. Weekly meetings were organized where annotators could share their findings and doubts and check their progress with us. The annotation process itself lasted for about three weeks with the annotators working part-time. An example of the annotation format and how this format was later converted into an XML format is described in Section 2.3.3.

2.3.1 Annotation of linguistic type

On the basis of an exploratory inspection of the data, we classified norm deviations into the following types:

Capitalization The text is capitalized for emphasis or emotive purposes, or else proper nouns are not capitalized: “y NO es broma” [NOT kidding] for “y no es broma” or “me recorro españa” [I go around spain] for “me recorro España”.

Accentuation Graphical accents are omitted: “en numeros rojos” for “en números rojos” [in the red].

Punctuation Punctuation signs are omitted or reduplicated; this includes also omission of blank spaces: “Te quiero!!!!!!!” for “¡Te quiero!” [I love you!], “aver” for “a ver” [let’s see].

Informal Spelling All systematic shortcuts and character substitutions intentionally made by the user: “pq” for “porque” [because], “t kiere muxo” for “te quiere mucho” [he loves you so much].

Spelling errors All spelling errors not included in the previous categories, including conventional misspellings, such as “oie” for “oye” [listen] or “targetas” for “tarjetas” [cards]; typos,

such as “dicindo” for “diciendo” [saying]; and intentional or unintentional reduplication of characters: as in “cooordenadas” for “coordenadas” [coordinates] , “alistarmeeeee” for “alistarme” [join up] or “frrrrrío” for “frío” [cold].

Other errors (lexical, syntactic): e.g., agreement errors or missing prepositions: “delante mi casa” for “delante de mi casa” [in front of my house], “mucho gente” for “mucha gente” [lots of people].

Arguably some of the above categories could have been divided into finer-grained ones. For instance, the two types of capitalization phenomena that we exemplify are quite distinct in nature: one can be attributed to an intentional communication goal to emphasize some idea, the other is rather a relaxation of standard spelling norms. Similarly, spellings like “frrrrrío” could be included in the category informal spelling, instead of spelling error. However, we could not distinguish the reduplication of letters owing to slips of the keyboard – presumably like “cooordenadas” – from those due to an intentional communication goal. We decided to keep the annotation process as simple as possible at this stage of the research – see also Brody and Diakopoulos (2011), a study exclusively devoted to the analysis and exploitation of letter reduplication for sentiment analysis of Twitter, which suggests that the semantic interpretation of certain UGC characteristics is a task in itself.

2.3.2 Annotation of transformation operations

Annotations included also the type of transformation operation with respect to the normalized version that a given deviation involves. The transformation operations we adopted are derived from Damerau (1964):

Addition A character or word is added, as in ‘dijo *de* que vendría’ for ‘dijo que vendría’.

Omission A character or word is omitted, as in “di*ci*ndo” for “dic*ien*do” or as in “delante mi casa” for “delante *de* mi casa”.

Substitution One or more character or words are replaced by others, as in “t *k*iere mu*x*o” for “te *qu*iere mu*ch*o”, where ‘k’ replaces ‘qu’ and ‘x’ replaces ‘ch’.

Transposition One character or word is placed before or after the expected position, as in “*ua*tobús” for “*au*tobús”

Duplication A character or a word is repeated one or several times, as in “frrrrrío” for “frío”.[4]

2.3.3 Annotation format

With the instructions described in Section 2.3, annotators would correct a fragment like the one shown in (1) into something like the fragment shown in (2).

(1) (...) estéticamente a la mayoria dela gente no gusta por su forma (...)
(2) (...) estéticamente a la [1 mayoría] [1 de la] no gusta por su forma (...)

Manual annotations were then automatically extracted and a table was generated. In a second step, they were manually classified by annotators into one of the categories described above. As a result, we obtained a table of error-correction pairs like the ones exemplified in Table 2.

[4] This is admittedly a subset of the deviations accounted for by addition transformation operations, but its high frequency and relevance in UGC texts justifies a category in its own.

Deviation	Correcction	Ling. Type	Transform. type
mayoria	mayoría	Acc	Subst
dela	dela	Punct	Om
q	que	InfSpl	Om
∅	la	Other	Om
diviertansee	diviértanse	Acc	Subst
diviertansee	diviértanse	SplErr	Add
(..)			

Table 2. *Table for deviation categorization completed by annotators in Step 2 of the annotation process.*

```
<devs start="1" offset="11" norm="Con certeza">
  <dev type="Punctuation" transf="BlankMissing"/>
  <dev type="Capitalization" transf="Substitution"/>
</devs>

<devs start="12" offset="30" norm="amor =)">
  <dev type="Punctuation" transf="BlankMissing"/>
  <dev type="SpellingError" transf="Reduplication"/>
</devs>
```

Fig. 2. XML annotation sample.

The initial manual annotation was mapped into an XML annotation scheme. This scheme is scalable and compatible with the Text Encoding Initiative (TEI) conventions. It is conceived as a stand-off annotation: instead of mixing the data with the metadata, the original text is preserved as-is while the annotation forms a separate layer, linked to the original text through offset indicators.

Figure 2 shows the annotation for the sentence "concerteza amooor=)", a real Twitter message. The normalized version is "|Con| |certeza| |amor| |=)|", where the three original tokens, including a space, result in four normalized tokens, including three spaces. The word 'con' appears capitalized – because it is the beginning of a sentence – and the word 'amooor' is reduced to 'amor'. Both instances of annotation show how multiple annotation of deviation types is possible, since more than one $<dev>$-tag can be assigned to the same normalized sequence of characters.

2.4 Distribution of deviation types across the different text sources

Overall, the rate of deviated input in our UGC reference corpus is quite high: over a fifth of the word types (20.8%) contain some error or deviation. This rate varies depending on the types of texts ranging from 4.62% in more edited text, such as blog posts, to over 25% in Twitter and informal consumer reviews. Note that this percentage is calculated on word types, not on word instances. The actual rate of error in word instances, as computed on the total number of words in the corpus, is around 5.7%.

Table 3 shows the percentage of word types that were manually corrected with respect to the total number of word types in each corpus, classified by error or deviation type. With minor exceptions, the frequency distribution of the deviation types does not exhibit significant variations across the different text sources.

Due to their relative frequency, three types of deviations clearly stand out over the rest: spelling errors, capitalization and accentuation. Even though the "Spelling errors" class in our classification includes reduplications with expressive or emotive purposes, most of the instances are "ordinary" orthographic errors, which, together with accentuation problems are well handled by conventional spell checkers. This fact has partly motivated the strategy we have chosen to deal with UGC text normalization, as we explain in Section 4.

	SplErr	Cap	Acc	Punct	InfSpl	Other	Total
Twitter	8.11	7.29	6.25	1.77	1.52	0.68	25.62
Blogs	2.64	1.38	0.41	0.12	0.01	0.06	4.62
Booking	3.71	1.71	8.71	0	0.57	1.56	16.26
CIAO-Banking	4.98	12.34	9.68	0.35	0.14	0.08	27.57
CIAO-CARS	8.95	5.47	7.99	1.86	0.28	0.50	25.05
CIAO-MOBILE	5.70	10.84	9.78	1.47	0.68	0.93	29.4
Total (cross-domain)	6.31	6.30	6.08	1.11	0.57	0.43	20.8

Table 3. *Percentage of deviations (in terms of word-types) according to its linguistic type, across genres.*

2.4.1 Characteristics of Spanish UGC and English UGC

There is relatively little research on the characterization of the specificities of UGC. However, Sproat et al. (2001) and Han and Baldwin (2011) offer some interesting information. Sproat et al. (2001) find that the percentage of "non-standard words" (NSW) in newspaper text is 8.8%, 43.4% in real state ads, 27.3% in servlist emails on the topic palmtop computers, and 22% in servlist emails on cooking recipes. Sproat et al. (2001) include in the NSW category phenomena such as abbreviations (*adv* for advertisement, *N.Y.* for New York, or *gov't* for government), any textual items including numbers (from phone numbers to car models), non-spoken elements (certain punctuation or token boundaries), and spelling errors or funny spellings (sic).

Han and Baldwin (2011, p. 370–371) provide information on the OOV rate found in newspaper text, SMS and Twitter text. Their findings show that both Twitter and SMS present a larger percentage of OOV words independent of the size of the percentage of the corpus looked at, while newspaper text presents a more Zipfian distribution of OOV words – that is, the more text you look at, the fewer newer or unknown words are found. One other interesting finding is that OOV words in SMS tend to be personal names. Last but not least, Han and Baldwin find also that the most frequent reason for "ill-formed" words are missing or extraneous letters, 72.44%, while

the second most frequent reason is "slang" – words found in a slang dictionary available on the Internet.

Though our findings are not fully comparable to those in the two previous references, we can see that our blog corpus is the one that presents the lowest deviation rate – comparable to newspaper text if we take into account that Sproat et al. (2001) were looking to a wider variety of non-standard forms. In contrast, our other corpora present very high rates of deviations – which are in line with the findings of both Sproat et al. (2001) and Han and Baldwin (2011) in their less formal types of texts.

3 Processing UGC text

Our hypothesis is that the percentage of deviations present in UGC text will have an impact on the performance of standard NLP tools. In a similar experiment, Foster (2010) detects problems with the handling of long coordinated sentences (mainly in the presence of erratic punctuation usage), domain-specific fixed expressions and unknown words.

To gauge the impact of deviations on the linguistic processing of UGC text, we processed the original version of the corpus and the manually corrected one, using a conventional linguistic processing pipeline for Spanish (Rodríguez et al., 2010). The pipeline consists of general-scope, state-of-the-art linguistic tools, not specifically adapted to UGC text, integrated on a UIMA platform.

We then compared the outcome in terms of changes in the resulting annotation. According to our analysis, the differences in the results of processing the two versions amount to 30% to 100% of the tokens, with a varying percentage depending on type of error, task and domain, as we see below.

3.1 Impact on lexical coverage

As we had expected, the normalization of the input increases lexical coverage (see Gouws et al. (2011) on the effects of normalization on lexical coverage). Table 4 shows the percentage of words – in terms of individual instances and word types – covered by the system's lexical resources in the original and in the manually normalized version. These values are notably lower for Twitter than for the rest of the sources.

The increase in coverage after normalization is shown between brackets. This increase turns out to be more evident in the comparison of word types than in the comparison of word instances, perhaps as a side effect: The normalization of deviated forms is actually decreasing the number of hapax (words occurring only once).

3.2 Impact on the performance of three basic NLP processing tasks

We investigated the effect of normalization on three basic NLP processing tasks: (i) lemmatization, (ii) part-of-speech tagging (short-tag or syntactic category), and (iii) assignment of morphosyntactic features (gender, number, tense...). These tasks are at the root of more complex or higher level processing tasks, and errors at this level are likely to propagate and affect other tasks such as constituent analysis, dependency relation identification, NERC, and other NLP tasks (see for instance Grefenstette and Tapanainen (1994), Chung and Gildea (2009) or Mohamed (2011)).

		Original	Normalized
Twitter	I	81.3	83.7 (+2.4)
	T	65.6	68.8 (+3.2)
Blog	I	95.5	96.3 (+0.8)
	T	86.6	89.0 (+2.4)
Booking	I	97.4	98.9 (+1.5)
	T	92.0	96.2 (+4.2)
Ciao	I	95.4	96.8 (+1.4)
	T	80.4	85.4 (+5.0)

Table 4. *Percentage of word coverage (instances, I, and types, T) in both the original and the corrected versions.*

	Lemmatization	PoS tagging	Morphosynt. features
Blog	94.6	43.8	62.6
Booking-Ciao	88.6	43.7	65.5
Twitter	85.4	49.8	64.5
Total corpus	89.6	45.8	64.2

Table 5. *Percentage of deviating words incorrectly analyzed, by domain.*

Table 5 shows the percentage of words for which a change in the resulting linguistic analysis is found after normalization [5] . In general, results are fairly uniform across the three genres. On the one hand, we observe that lemmatization is very sensitive to the presence of error. The percentage of wrong lemma assignment to erroneous, or non-normalized, word forms is 90% overall, and between 85% and 95% depending on the corpus.

On the other hand, assignment of syntactic category is generally quite robust to deviation, since only less than half of the non-normalized word forms (around 46%) change their category assignment after having been corrected or normalized, while the proportion of wrong assignment of morphosyntactic features is a little higher, around 64%, across genres.

Table 6 presents the same information broken down by deviation type, for the four most frequent types. Capitalization turns out to be the least detrimental across these basic tasks, while accentuation is particularly harmful, even for a robust task such as PoS tag assignment (main grammatical category).

As we can see in Table 6, lemmatization is likely to be inaccurate for most deviated words.

[5] The NLP tools used for the analysis are state-of-the-art and provide reasonable results, therefore any change in these results as a consequence of manually normalizing the input can be safely assumed to be positive, or at least neutral. Manual verification of a random sample showed only positive changes.

	Lemmatization	PoS tagging	Morphosynt. features
Capitalization	68.3	29.5	56.3
Accentuation	97.8	65.2	75.9
Spelling errors	97.9	56.0	71.1
Punctuation	99.8	46.1	59.1

Table 6. *Percentage of deviating words incorrectly analyzed, by type of error.*

These results are consistent with previous findings in the field, where the analysis of lexical items was found to be particularly sensitive (e.g., Han and Baldwin, 2011, p. 370–371, Gouws et al., 2011, p. 28). Lemmatization errors increase the variability for the same concept and thus are likely to affect most semantic related tasks.

4 Building a normalizer on top of an existing spell checker

The large presence of deviated forms in UGC text and its costly impact on the performance of NLP tools convinced us of the need for a specific solution that addresses the problem of processing of this type of text. As discussed in Section 1, we find two approaches dealing with this issue in the literature: either transform the input text (i.e. normalize it) or adapt the tools (e.g. by retraining the models). While the second option is feasible mostly for statistically trained tools (which is the case for Foster (2010)), the first option should work for any tool, statistical and rule-based, and it is the course we have followed in this work.

Among the different approaches to normalization presented in Section 1 we have chosen to view normalization as a spell checking task, particularly motivated by the high rate of typical orthographic errors (including accentuation) in UGC text. For this purpose we have built an automatic normalizer on top of the Spanish version of COTiG, a spell and grammar checker first developed for Catalan (Quixal et al., 2008).

A key difference between a regular spell checker and a normalizer is interactivity with an end-user. The lack of interactivity in the normalization task has an important implication: a specific strategy has to be put in place to rank possible correction candidates and decisively choose the best one over the whole set. The rest of the article deals with the development of a dedicated **selector module** that ranks the list of correction candidates proposed by the spell checker and selects the highest ranked one.

There are other important aspects that need to be addressed to effectively turn a regular spell checker into a normalizer. A main concern is the rate of false positives. If over-correcting may be annoying for the user of a spell checker, over-normalizing, i.e. introducing unwanted changes to the original text, can be invalidating for a normalizer. Out of vocabulary words are one of the main sources of false positives. A straightforward way to deal with this is using larger dictionaries or dictionaries adapted to the domain in question. Moreover, typical UGC phenomena, such as informal spellings or emoticons may not be appropriately dealt with by using standard spell checking procedures, such as edit distance algorithms; thus, ad hoc or specific strategies to approach these phenomena may also be required.

5 Selection of the right candidate using language models

The output of the spell checker consists of an unordered list of correction candidates obtained through the application of its own algorithms for the generation of correction proposals, based on simple editing-distance criteria. For instance, given a sentence such as *aunque mire otros coxes de la misma categoría* [even if I look at other cars of the same type], the list generated by the spell checker for the word *coxes* (*coches* [cars] intended) looks as shown in Figure 3.

```
<devs begin="318" end="322" original="coxes">
    <proposals>
      <proposal id="1" > boxes  </proposal>
      <proposal id="2" > comes  </proposal>
      <proposal id="3" > coses  </proposal>
      <proposal id="4" > coches </proposal>
      <proposal id="5" > coxas  </proposal>
      <proposal id="6" > coges  </proposal>
      <proposal id="7" > corres </proposal>
      <proposal id="8" > coxis  </proposal>
      <proposal id="9" > coles  </proposal>
      <proposal id="10"> boches </proposal>
      <proposal id="11"> coces  </proposal>
    </proposals>
  </devs>
```

Fig. 3. Output of the normalizer in XML format containing a list of correction candidates.

To select the most probable correction among a set of candidates, we experimented with the use of different trigram models trained on an in-domain corpus described in Section 5.1, each model conveying a different level of information.

Inspired by the so-called factored models employed in statistical machine translation (Koehn and Hoang, 2007; Bilmes and Kirchhoff, 2003), our selector module enables the integration of different language models trained on different automatically generated linguistic annotations at the word-level:

- Truecase form model (TC). This model was trained on the original unmodified text, where uppercase and lowercase instances of the same form are different words. It is the most specific – or the least abstract one – since it is the model with the largest vocabulary.
- Lowercase form model (LC). This model has been trained on the lowercase version of the original corpus. Uppercase and lowercase versions of the same form are now the same word.
- Lemma model (Lemma). This model has been trained on the lemmatized version of the original corpus, where each inflected form has been substituted by its root or lemma. Plural and singular versions of the same noun are now the same word; the same happens with variations in person, tense and number of verbal forms.
- Part-of-Speech model (PoS). This model has been trained on the PoS tags version of the corpus. Tags are PAROLE-style part-of-speech long tags, which for each word include its syntactic category and morphosyntactic features (e.g. AQ0CP0, NCFP000, VSIC3P0; for

details see (Villegas et al., 1996)). This model is the most general – or the one that abstracts at a higher level – since it has the smallest vocabulary.

5.1 Building the models

The corpus used to build the models is a 24 million word corpus collected from the Web that comprises texts from the same domains and genres included in the reference corpus, namely: banking, cars, mobile, Twitter and blogs.[6]

The lemma and morphosyntactic labels were assigned using our in-house linguistic pipeline (cf. Section 3). We then built 4 trigram models with the IRSTLM toolkit (Federico et al., 2008) using the "modified shift-beta" as smoothing method, also known as "improved Kneser-Ney smoothing".

5.2 Querying the models

At run time we query the models using a sliding window over the proposed correction of at most 5 words. For the example in Figure 3, 11 different 5-word candidate strings are generated, one for each candidate correction, surrounded by its immediate context.

```
S1  = mire otros boxes  de la
S2  = mire otros comes  de la
S3  = mire otros coses  de la
S4  = mire otros coches de la
S5  = mire otros coxas  de la
S6  = mire otros coges  de la
S7  = mire otros corres de la
S8  = mire otros coxis  de la
S9  = mire otros coles  de la
S10 = mire otros boches de la
S11 = mire otros coces  de la
```

For each candidate strings S_i, we generate 4 parallel strings:

- S_{TC} with all words in truecase: *mire otros boxes de la*
- S_{LC} with all words in lowercase: *mire otros boxes de la*
- S_{Lemma} with lemmas only: *mirar otro box de la*
- S_{PoS} with PoS only: *VMSP3S0 DI3MP0 NCMP000 SPS00 DA3FS0*

With $C(M, S)$ being the cost of string S (i.e. logarithm of the probability) according to model M, we query each of the four models for the cost value of each one of the strings. For example, $C(TC, S_{TC})$ is the cost of the truecase string computed against the TC model. That means that for each candidate, we get four cost values: $C(TC, S_{TC})$, $C(LC, S_{LC})$, $C(Lemma, S_{Lemma})$ and $C(PoS, S_{PoS})$.

[6] Much larger data sets are available, however recent research has shown that selecting only a fraction (e.g. less than 10%) of the data according to the closeness to the domain can be as effective as keeping the whole data set. Several authors select pseudo in-domain language model training data with cross-entropy methods and obtain good results in perplexity or in a machine translation task (Moore and Lewis (2010); Axelrod et al. (2011); Rousseau et al. (2011); Toral (2013)). We thus focused our effort in collecting in-domain data.

5.3 Combining the models

The aim of building the four different models is to evaluate which of them is more discriminative at ranking the different candidates. We also experiment with the combination of models to find out whether such a strategy can improve over the results of the best performing model used in isolation. Eq. (1) shows the linear equation for the parametrization of the combination of the different models.

$$\begin{aligned} C(total, S) = \lambda_{TC} C(TC, S_{TC}) + \lambda_{LC} C(LC, S_{LC}) + \\ \lambda_{Lemma} C(Lemma, S_{Lemma}) + \lambda_{PoS} C(PoS, S_{PoS}) \end{aligned} \tag{1}$$

5.3.1 Linear interpolation of language models

Linear interpolation of language models is one of the most popular model combination techniques. Given a set of n-gram language models, linear interpolation consists in computing the weighted average of the component model costs as shown in Eq. 1. The interpolation weight λ_i, where $\{i = TC, LC, Lemma, PoS\}$ satisfies $\sum_i \lambda_i = 1$ and it is typically tuned to optimize the perplexity of the development set. We empirically adjusted the combination weights by following the standard procedure for linear interpolation in language modeling in order to build a unique optimized model. This procedure uses the expectation-maximization (EM) algorithm to minimize the perplexity over a specific development set (Stolcke, 2002). To apply this procedure, we previously calculate the probability of the same random variable, which in the TC model is a word and in the other three models (LC, Lemma and PoS) is a class. For a class model C, we calculate:

$$P_C(\omega_n | \omega_{n-k+1} ... \omega_{n-1}) = \sum_i P(\omega_n | c^i_{\omega_n} | c_{\omega_{n-k+1}} ... c_{w_{n-1}}) \tag{2}$$

where $c^i_{\omega_n}$ are different classes of the word ω_n. Using this equation for the three models where the variable is a class, we can use EM to calculate the mentioned interpolation weights λ_i, where $\{i = TC, LC, Lemma, PoS\}$.

The development set was extracted out of the text of the reference corpus (described in Section 2), as detailed in Section 5.3.2. This development set of around 50K words was used in the optimization phase. Table 7 shows the vocabulary sizes for our training and development sets for the four models: TC, LC, Lemma, and PoS. Even though the number of words is the same for the four training sets – i.e. 24 million words –, the size of the respective vocabularies varies considerably, going from 525,742 in the case of the TC model to only 243 in the case of the PoS model. The same applies to the development set.

The language model weight optimization was computed using the *compute-best-mix* function of the SRILM toolkit (Stolcke, 2002). We used the option of *expand-classes* to normalize the vocabularies in classes to words (for LC, PoS and Lemma language models). The perplexity of each language model is shown in Table 8. The PoS language model is the one with the highest perplexity, which seems reasonable because it has fewer classes and, therefore, a higher degree of ambiguity. In contrast, the TC model, where each word is a class, has the lowest perplexity.

The resulting optimization assigns a sizeable weight to the truecase model, a small weight to

	TC	LC	Lemma	PoS
Training	525742	456219	440818	243
Development	12673	11482	9058	184

Table 7. *Vocabulary size (unigrams counts) for the training and development corpus.*

	Perplexity
Truecase	94.5662
Lowercase	103.364
Lemma	293.121
PoS	504.553
Combination	92.9201

Table 8. *Perplexity in the development set for each language model*

the lowercase model and practically overlooks the lemma and part-of-speech models. The final set of weights is presented in Eq.(3):

$$\begin{aligned} C(total, S) = 0.961861 \ * \ C(TC, S_{TC}) \ + \ 0.0309452 \ * \ C(LC, S_{LC}) \\ + \ 0.00690778 \ * \ C(Lemma, S_{Lemma}) \ + \ 0.00000003 \ * \ C(PoS, S_{PoS}) \end{aligned} \tag{3}$$

Note that this is theoretically a methodology to optimize the weights over language models with different word classes. However, in practice, the task of expanding classes may be too ambiguous, which directly affects the interpolation procedure. For example, when expanding the lowercase word class of *zapatero* we have only two possible surface forms *zapatero* and *Zapatero*, which is quite an easy task. But in the case of expanding the PoS word class of *Prepositon*, we have 23 possible word surface forms. At the end, the weight given to the language model with the most generic classes (in this case PoS) is quite low, which may be a direct consequence of an improper expansion of classes.

5.3.2 Optimization on the task

The combination of the models can be further optimized by tuning the parameters of the four language models to minimize errors *directly* on the selection task. Optimization on the task is possible if a gold standard can be used to iteratively measure improvement. For this we used the reference corpus described in Section 2. We processed the original non-corrected section of the reference corpus, and found a total of 4235 deviation annotations for which the underlying spell checker correctly detects an error *and* the normalization of reference is within the set of candidates proposed by the selector module. Therefore, the number of deviations that we are going to optimize in this experiment is the number of deviations (correctly detected by the

spell checker) for which the candidate ranked first by the selector module coincides with the normalization of reference according to our gold standard. To evaluate our results, we use the manually normalized reference corpus. We divide it in two parts: 30% for the development set (the same one as for the linear interpolation) for optimizing on the task and 70% for evaluation.

To iteratively minimize errors we used a modified version of the Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 1992; Lambert and Banchs, 2006) as an optimization algorithm.

SPSA Algorithm. The SPSA procedure is in the general recursive stochastic approximation form, as shown in Eq. (4):

$$\hat{\lambda}_{k+1} = \hat{\lambda}_k - \mathbf{a}_k \hat{\mathbf{g}}_k(\hat{\lambda}_k) \tag{4}$$

where $\hat{\mathbf{g}}_k(\hat{\lambda}_k)$ is the estimate of the gradient $\mathbf{g}(\lambda) \equiv \partial E/\partial\lambda$ at the iterate $\hat{\lambda}_k$, and a_k denotes a positive number that usually decreases as k increases. The gradient was computed with a one-sided approximation which, given $E(\hat{\lambda}_k)$, requires the evaluation of $E(\hat{\lambda}_k + \text{perturbation})$. In the simultaneous perturbation approximation, all elements of $\hat{\lambda}_k$ are randomly perturbed together[7] and the approximated gradient vector is:

$$\hat{\mathbf{g}}_k(\hat{\lambda}_k) = \frac{E(\hat{\lambda}_k + c_k\mathbf{\Delta_k}) - E(\hat{\lambda}_k)}{c_k} \begin{bmatrix} 1/\Delta_{k1} \\ 1/\Delta_{k2} \\ \vdots \\ 1/\Delta_{kN} \end{bmatrix} \tag{5}$$

In Eq. 5, $\mathbf{\Delta}_k$ is a perturbation vector of the same dimension N as λ, whose values Δ_i are computed randomly. c_k denotes a small positive number that decreases as k increases. We took for each component Δ_i of $\mathbf{\Delta}_k$ a Bernoulli ± 1 distribution with probability of $1/2$ for each ± 1 outcome, or a $0, \pm 1$ distribution with probability of $1/3$ for each outcome. Full details of our implementation of the algorithm are given in Lambert and Banchs (2006). The main change with respect to the original algorithm is that the parameters are not necessarily updated at each iteration. If the new set of parameters $\hat{\lambda}_{k+1}$ is worse than the current one, it is updated with a probability which depends on how much worse it is. The algorithm typically converged after 50 to 60 iterations. Note that, in general, SPSA converges to a local maximum.

Task optimization. Figure 4 exemplifies the optimization process for the candidate selection task with the goal of maximizing precision (minimizing the number of times the correct proposal was not ranked first in the list). With the initial set of weights, $\hat{\lambda}_0$, we normalize the original part of the development corpus and calculate the precision against the manually corrected reference of the same corpus. This gives us $E(\hat{\lambda}_0)$. We perturb the $\hat{\lambda}_0$ vector with $c_0\mathbf{\Delta_0}$, normalize the development corpus with the perturbed set of weights and calculate its precision, which gives $E(\hat{\lambda}_0 + c_0\mathbf{\Delta_0})$. We can now update the set of weights according to Eqs. (4) and (5), giving $\hat{\lambda}_1$. We evaluate $E(\hat{\lambda}_1)$ and according to the result, we take $\hat{\lambda}_1$ or keep $\hat{\lambda}_0$ as the current set of weights. Then we perturb again the current set of weights with the $c_k\mathbf{\Delta_k}$ vector, and so on.

[7] Compared to a finite-difference gradient approximation, involving N times more function evaluations, the simultaneous approximation causes deviations of the search path. These deviations are averaged out in reaching a solution and according to Spall (1998), under reasonably general conditions, both gradient approximations achieve the same level of statistical accuracy for a given number of iterations.

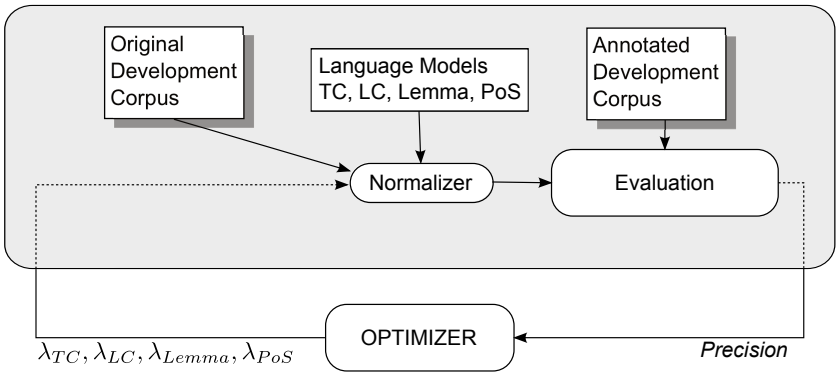

Fig. 4. Language model weight optimization according to the selection precision.

The equation resulting from the optimization assigns equal moderate weight to TC and LC, a small weight to the Lemma model and zero to the PoS model, as shown in Eq.(6):

$$\begin{aligned} C(total, S) = 0.4519 \ * \ C(TC, S_{TC}) \ + \ 0.4519 \ * \ C(LC, S_{LC}) \\ + \ 0.0962 \ * \ C(Lemma, S_{Lemma}) \ + \ 0 \ * \ C(PoS, S_{PoS}) \end{aligned} \tag{6}$$

5.4 Evaluation of candidate selection

As indicated in the previous section, we used for evaluation purposes the unused portion of the annotated reference corpus, i.e. 70% of the total. We tested each of the four models individually and the two different model combinations presented in Section 5.3: linear interpolation and optimization on the task. As a baseline we ran the normalizer without the selector module, i.e., taking the first item in the list as proposed by the underlying spell checker – which has an almost non-existent ranking strategy (Quixal et al., 2008). Moreover, since Twitter text is known to present specific features that make it different from other UGC text and that may affect performance, we also evaluate Twitter and the rest of the corpus sources separately. We repeated the experiment of optimization on the task with specific development and test sets for Twitter and non-Twitter.

The results, in terms of percentage of instances where the reference correction was ranked first, are shown in Table 9. The model built on a lowercase version of the corpus turns out to be the more discriminating model in isolation, with a precision of 86.59%, as measured on the test set (second column of the table). On the same set, we see that neither model combination improves the precision achieved by the LC model, although they perform better than the rest of the models used in isolation. Noticeably PoS obtains results that are below the baseline. Both model combination techniques, Linear Interpolation and Optimization on the task, obtain quite comparable results on the test set, although the results of the latter are slightly better (around 0.4 points improvement). The first column of Table 9 shows the results for the development set of the corpus, used to obtain the optimized set of weights described in Section 5.3.2. We observe that in this case, the weighted combination obtained by the Optimization on the task technique improves on the result of the best performing model in isolation, i.e. LC, in the development set, but not

Model	Precision	
	Dev	Test
LC	88.31	86.59
Optimization on the task	89.12	85.93
Linear interpolation	86.76	85.53
TC	86.76	84.99
Lemma	73.63	69.94
Baseline	51.04	61.99
PoS	62.66	52.50

Table 9. *Precision values for each model, their combination and baseline.*

Model	Precision	
	Twitter	non-Twitter
LC	82.56 (-4.03)	87.84 (+1,25)
Optimization on the task	80.12 (-5.81)	87.46 (+1.53)
Linear interpolation	79.81 (-5.72)	86.93 (+1.40)
TC	79.51 (-5.48)	86.70 (+1.71)
Lemma	67.12 (-2.82)	69.97 (+0.03)
Baseline	56.88 (-5.11)	63.58 (+1.59)
PoS	55.35 (-2.85)	49.97 (+3.03)

Table 10. *Precision values for each model, their combination and baseline, separated results for Twitter and non-Twitter text, from the test set.*

in the test set and that Linear Interpolation[8] and the TC model achieve the same precision. Remarkably the baseline system (without a selector module) obtains a much lower result on the development set (10 points below the test set), and in this case it is clearly surpassed by the worst performing model, i.e. PoS. Table 10 shows separated results for Twitter and non-Twitter text. In this table we put between brackets the negative or positive difference of each value with respect to the values in Table 9. All in all the precision measure of the correction selection drops 4.5 points on average when tested on Twitter text. The ranking of the models does not change for the separated results.

[8] Recall that the linear interpolation was tuned on a different development set, described in Section 5.3.1.

5.5 Discussion of accuracy, precision and recall and related work

The research that is closer to the experiment presented in this paper is that by Han and Baldwin (2011) and Liu et al. (2012), since they both work on data including text other than Twitter text, and they both work on the normalization of domain-unspecific text – Liu et al. (2012) call their approach a broad-coverage system.

In terms of spell checking detection and generation of proposals both Han and Baldwin (2011) and Liu et al. (2012) work with systems generated by themselves, rather than using existing resources. Particularly Liu et al. (2012) present proposal generation strategies inspired by visual priming and in word similarity measures. With the former they can favor the generation and ranking of spelling corrections on the basis of their frequency in social text and the most economic transformation operation – the fewer characters that are changed, the better. With the latter they generate lists of representatives that one particular word can have in a text. For instance, for a word like *please* they can generate the set {*pleas, pleeas, pleaas, pleeaas, pleeeaas*}. Han and Baldwin (2011) use an algorithm based on morphophonemic similarity to generate correction proposals. These two approaches allow for higher accuracy in the percentage of errors for which a correction proposal is generated – not selected.

In addition to the differences in the proposal generation strategies, there is also an important difference, which is that we include the normalization of tokens as part of the normalization process, as opposed to what the above mentioned works do. For these reasons results are hardly comparable among the different experiments. With this caveat in mind, our restricted experiment, in which we are using only those cases where the spell-checker has been able to correctly detect an error and propose a set of correction candidates containing the reference correction, shows a higher precision in terms of 1st-ranked correct proposals.[9] Compare our 86.59% vs. Liu et al. (2012)'s 84.13% when using a Twitter-based language model or their 79.12% when using a web language model. Their recall is respectively 78.38% and 77.11%, while Han and Baldwin (2011) reach a precision and recall of 75.30%.

6 Conclusions

In this paper, we presented and analyzed a Spanish corpus of UGC including text from five different sources. We have seen that UGC text presents some particular features that set it apart from standard text. On the one hand, it contains specific phenomena that add expressiveness, such as emoticons, informal spellings, non-standard capitalization and reduplication. On the other hand, the rate of typical orthographic errors is much higher than in more edited types of text.

We analyzed the impact that norm deviations have on standard NLP processing tools, by comparing the results of processing it with the results of processing its manually corrected version. As a result, we observed that UGC text has a negative impact on the performance of three basic NLP tasks: lemmatization, assignment of the main syntactic category, and assignment of morphosyntactic features.

We propose to address the problem by normalizing UGC text before the linguistic processing. For this, we have used a conventional spell-checker on top of which we have built a module to

[9] The actual precision and recall numbers of the underlying, non-adapted spell-checker are quite low: 52% and 57% respectively.

automatically select the best correction candidate. For the selection task, we wanted to experiment with the use of linguistic information with various levels of fine-grainedness. Therefore, we have trained four language models on a large domain-specific corpus, each model containing a different degree of information in a trade off with its generalization capabilities. The model based on the lowercase version of the corpus has turned out to be the most predictive, with a reasonable precision value of 86.59% as measured on the test set. This figure is well above the baseline and is slightly higher than the highest accuracy obtained by related work of Liu et al. (2012) – 84.13% in the ranking of correction proposals in the first position of the list. We have also experimented with two methods for combining the four models: model optimization directly on the final task, using a modified version of the SPSA algorithm, and a linear combination of the models optimized independently of the task. Both methods achieve comparable results, although the weights obtained for each language model are quite different. This is an interesting conclusion since optimizing independently for the task is much easier than the other way round. Similar conclusions have been obtained in different tasks (e.g. in a phrase-based statistical machine translation). Results of the combination of models are close to the best performing model, i.e. LC, but do not perform better, at least on our test set of data. The evaluation for the additional experiment using only Twitter text shows a significant drop in precision, compared to non-Twitter text. The experiment of using different language models has been useful to obtain better results using the LC model instead of the baseline TC model.

7 Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. The authors would also like to thank Barcelona Media Innovation Center (where a part of the work was conducted) and Institute for Infocomm Research for their support and permission to publish this work.

The research leading to these results has received funding from the CDTI through the CENIT project Social Media and from the Seventh Framework Program of the European Commission through the International Outgoing Fellowship (IMTraP-2011-29951) and the Intra-European Fellowship (CrossLingMind-2011-300828) Marie Curie Actions.

References

Agarwal, A., Xie, B., Vovsha, I., Rambow, O., and Passonneau, R. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 30–38, Portland, Oregon. Association for Computational Linguistics.

Ahmed, F., Luca, E. W. D., and Nürnberger, A. 2010. Revised n-gram based automatic spelling correction tool to improve retrieval effectiveness. *Research journal on computer science and computer engineering with applications (Polibits) ISSN 1870-9044*, 40.

Alonso, L. 2010. Inisghts lingüísticos relativos a la normalización léxica de contenidos generados por usuarios. *Subjetividad y Procesos cognitivos*, 14(2):20–31. Printed ISSN: 1666-244X, electronic ISSN :1852-7310.

Aminian, M., Avontuur, T., Azar, Z., Balemans, I., Elshof, L., Newell, R., van Noord, N., Ntavelos, A., and van Zaanen, M. 2012. Assigning part-of-speech to Dutch tweets. In Melero, M., editor, *Workshop "NLP can u tag #user-generated-content ?! via lrec-conf.org"*. Language Resources and Evaluation Conference.

Axelrod, A., He, X., and Gao, J. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Bender, E. M., Morgan, J. T., Oxley, M., Zachry, M., Hutchinson, B., Marin, A., Zhang, B., and Ostendorf, M. 2011. Annotating social acts: Authority claims and alignment moves in wikipedia talk pages. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 48–57, Portland, Oregon. Association for Computational Linguistics.

Bermingham, A. and Smeaton, A. 2010. Classifying sentiment in microblogs: is brevity an advantage? In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1833–1836. ACM.

Bilmes, J. A. and Kirchhoff, K. 2003. Factored language models and generalized parallel backoff. In *Proceedings of HLT/NACCL*, pages 4–6.

Brill, E. and Moore, R. C. 2000. An improved error model for noisy channel spelling correction. In *Proc. 38th Annual Meet. of the Assoc. for Comp. Ling.*, pages 282–293, Hong Kong.

Brody, S. and Diakopoulos, N. 2011. Cooooooooooooooollllllllllll!!!!!!!!!!!!!! using word lengthening to detect sentiment in microblogs. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 562–570, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., and Basu, A. 2007. Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*, 10(3):157–174.

Chung, T. and Gildea, D. 2009. Unsupervised tokenization for machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, Singapore.

Church, K. and Gale, W. 1990. Poor estimates of context are worse than none. In *Third Darpa Workshop on Speech and Natural Language*, Hidden Valley, PA.

Clark, E. 2003. Pre-processing very noisy text. In *Proc. of Workshop on Shallow Processing of Large Corpora*.

Clark, E. and Araki, K. 2011. Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. In *PACLING*.

Cook, P. and Stevenson, S. 2009. An unsupervised model for text message normalization. In *CALC '09: Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78, Morristown, NJ, USA. Association for Computational Linguistics.

Damerau, F. J. 1964. A Technique for Computer Detection and Correction of Errrors. *Communications of the ACM*, 7:171–176.

Eisenstein, J. 2013. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Federico, M., Bertoldi, N., and Cettolo, M. 2008. Irstlm: an open source toolkit for handling large scale language models. In *Interspeech*, pages 1618–1621, Brisbane, Australia.

Foster, J. 2010. "cba to check the spelling": Investigating Parser Performance on Discussion Forum Posts. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 381–384, Los Angeles, California. Association for Computational Linguistics.

Foster, J., Çetinoglu, Ö., Wagner, J., Le Roux, J., Hogan, S., Nivre, J., Hogan, D., Van Genabith,

J., et al. 2011. # hardtoparse: Pos tagging and parsing the twitterverse. In *Proceedings of the Workshop On Analyzing Microtext (AAAI 2011)*, pages 20–25.

Gianfortoni, P., Adamson, D., and Rosé, C. P. 2011. Modeling of stylistic variation in social media with stretchy patterns. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, DIALECTS '11, pages 49–59, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gouws, S., Metzler, D., Cai, C., and Hovy, E. 2011. Contextual bearing on linguistic variation in social media. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 20–29, Portland, Oregon. Association for Computational Linguistics.

Grefenstette, G. and Tapanainen, P. 1994. [w]hat is a word, [w]hat is a sentence? [p]roblems of [t]okenization. In *Proceedings of the 3rd Conference on Computational Lexicography and Text Research*, pages 79–87, Budapest, Hungary.

Han, B. and Baldwin, T. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *ACL*, pages 368–378.

Hassan, H. and Menezes, A. 2013. Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1577–1586, Sofia, Bulgaria. Association for Computational Linguistics.

Henríquez Q., C. and Hernández, A. 2009. A n-gram based statistical machine translation approach for text normalization on chat-speak style communications. In *CAW 2.0 Workshop*.

Herring, S. 2012a. Discourse in Web 2.0: Familiar, reconfigured, and emergent. In Tannen, D. and Tester, A. M., editors, *Discourse 2.0: Language and new media*, Georgetown University Round Table on Languages and Linguistics 2011. Georgetown University, Washington, DC.

Herring, S. 2012b. Grammar and electronic communication. In Chapelle, C., editor, *Encyclopedia of applied linguistics*. Wiley-Blackwell, Hoboken, NJ.

Hodge, V. J. and Austin, J. 2003. A comparison of standard spell checking algorithms and novel binary neural approach. *IEEE Trans. Know. Dat. Eng.*, 15(5):1073–1081.

Kobus, C., Yvon, F., and Damnati, G. 2008. Normalizing SMS: are two metaphors better than one? In *COLING '08*.

Koehn, P. and Hoang, H. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876.

Kukich, K. 1992. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439.

Lambert, P. and Banchs, R. E. 2006. Tuning machine translation parameters with SPSA. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 190–196, Kyoto, Japan.

Liu, F., Weng, F., and Jiang, X. 2012. A broad-coverage normalization system for social media language. In *ACL (1)*, pages 1035–1044. The Association for Computer Linguistics.

Mangu, L. and Brill, E. 1997. Automatic rule acquisition for spelling correction. In *In Proceedings of the 14th International Conference on Machine Learning*, pages 734–741. Morgan Kaufmann.

Martins, B. and Silva, M. J. 2004. Spelling correction for search engine queries. In *EsTAL - España for Natural Language Processing*, Alicante, Spain.

Maynard, D., Bontcheva, K., and Rout, D. 2012. Challenges in developing opinion mining tools for social media. In Melero, M., editor, *Workshop "NLP can u tag #user-generated-content ?! via lrec-conf.org"*, Istanbul. Language Resources and Evaluation Conference.

Mays, E., Damerau, F. J., and Mercer, R. L. 1991. Context based spelling correction. *Information Processing and Management*, 27(5):517 – 522.

Michelson, M. and Knoblock, C. A. 2005. Semantic annotation of unstructured and ungrammatical text. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1091–1098.

Mohamed, E. 2011. The effect of automatic tokenization, vocalization, stemming, and POS tagging on Arabic dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 10–18, Portland, Oregon, USA. Association for Computational Linguistics.

Moore, R. C. and Lewis, W. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden. Association for Computational Linguistics.

Muñoz-García, O. and Navarro, C. 2012. Comparing user generated content published in different social media sources. In Melero, M., editor, *Workshop "NLP can u tag #user-generated-content ?! via lrec-conf.org"*, Istanbul. Language Resources and Evaluation Conference.

Pang, B. and Lee, L. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Quixal, M., Badia, T., Benavent, F., Boullosa, J. R., Domingo, J., Grau, B., Massó, G., and Valentín, O. 2008. User-Centred Design of Error Correction Tools. In Chair), N. C. C., Choukri, K., Maegaard, B., Mariani, J., Odjik, J., Piperidis, S., and Tapias, D., editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Ritter, A., Clark, S., Mausam, and Etzioni, O. 2011. Named entity recognition in tweets: An experimental study. In *EMNLP*, pages 1524–1534. ACL.

Rodríguez, C., Banchs, R., Codina, J., and Grivolla, J. 2010. Cometa: Semantic exploration of customer reviews to extract valuable information for business intelligence. Technical report, Barcelona Media Innovation Center.

Rousseau, A., Bougares, F., Deléglise, P., Schwenk, H., and Estève, Y. 2011. Lium's systems for the iwslt 2011 speech translation tasks. In *International Workshop on Spoken Language Translation*, San Francisco (USA).

Spall, J. C. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. Automat. Control*, 37:332–341.

Spall, J. C. 1998. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins APL Technical Digest*, 19(4):482–492.

Sproat, R., Black, A. W., Chen, S. F., Kumar, S., Ostendorf, M., and Richards, C. 2001. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333.

Stolcke, A. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286.

Toral, A. 2013. Hybrid selection of language model training data using linguistic information and perplexity. In *Proceedings of the Second Workshop on Hybrid Approaches to Translation*, pages 8–12, Sofia, Bulgaria. Association for Computational Linguistics.

Toutanova, K. and Moore, R. C. 2002. Pronunciation modeling for improved spelling correction. In *Proc. 40th Annual Meeting of the Assoc. for Comp. Ling.*, pages 144–151, Hong Kong.

Villegas, M., Brosa, M. I., and Bel, N. 1996. El léxico PAROLE del español. In *XIV Congreso de la Sociedad Española para el Procesamiento del Lenguaje.*

Zhu, C., Tang, J., Li, H., Ng, H. T., and Zhao, T. 2007. A unified tagging approach to text normalization. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 688–695, Prague, Czech Republic. Association for Computational Linguistics.