LYSGROUP: Adapting a Spanish microtext normalization system to English.

Yerai Doval, Jesús Vilares, Carlos Gómez-Rodríguez
LYS group, Departamento de Computación, Facultade de Informática,
Universidade da Coruña, Campus de Elviña, 15071 A Coruña, Spain
{yerai.doval, jvilares, cgomezr}@udc.es – www.grupolys.org

Abstract

In this article we describe the microtext normalization system we have used to participate in the Normalization of Noisy Text Task of the ACL W-NUT 2015 Workshop. Our normalization system was originally developed for text mining tasks on Spanish tweets. Our main goals during its development were flexibility, scalability and maintainability, in order to test a wide variety of approximations to the problem at hand with minimum effort. We will pay special attention to the process of adapting the components of our system to deal with English tweets which, as we will show, was achieved without major modifications of its base structure.

1 Introduction

The value of Twitter and other *microblogging* services as information sources in domains like marketing, business intelligence, journalism, etc. is obvious nowadays. Nevertheless, such amount of information can only be appropriately exploited through *text mining* techniques.

However, there are notable differences between "standard" language and the so-called *texting* used in those *microtexts*. In this kind of writings, it is important to reduce the number of characters used to fit their length restrictions while maintaining the readability of the message to some extent. To achieve this, most of the techniques applied rely on phonetics, thus being language-specific (López Rúa, 2007). For example: intentionally ignoring orthographic and grammar rules, as in "be like" for "am/is/are/was/were like" in the case of English or "asique" for "así que" in the case of Spanish; the usage of shortenings, contractions and abbreviations such as "c u" for "see you" in English or "ksa" for "casa" in Spanish; or the employment of *smileys* to express emotions, for instance :) to express happiness. These resulting terms are called *lexical variants* (Han et al., 2013).

The problem is that, in general, text mining tools are very sensitive to those phenomena, as they are designed for dealing with standard texts. Therefore, it is necessary to *normalize* these texts before their processing, that is, to transform them into standard language. This way "c u nxt week", for example, would be transformed into "see you next week". This is the goal of the W-NUT 2015 Normalization Task (Baldwin et al., 2015).

The rest of this paper is organized as follows: Section 2 describes the core architecture of our system, and how it was adapted to fit this shared task, and Section 3 presents the resources used. Next, Section 4 evaluates the system and discusses the results obtained. Finally, Section 5 presents our conclusions and considers some possible future improvements for our system.

2 Architecture

Our tweet normalization system was developed taking as basic premises its flexibility, scalability and maintainability. As a starting point, we took a previous prototype for Spanish tweet normalization (Vilares et al., 2013) which, although fully functional, did not turn out to be as flexible and maintainable as expected. This could have become a problem for future developments, since the adaptation effort needed to integrate new techniques would have been too large, so we decided to refactor the whole system to solve this.

The general scheme of the original system mimics that of Han and Baldwin (2011) and comprises three stages:

1. Tweet preprocessing.

2. In-vocabulary word identification (IV), based on the lexicon of the system, obtaining as

Proceedings of the ACL 2015 Workshop on Noisy User-generated Text, pages 99–105,
Beijing, China, July 31, 2015.

a result an initial set of out-of-vocabulary words (OOV).

3. OOV set processing in order to distinguish between correct words which are out of the system lexicon and proper *lexical variants*, obtaining for each one of the latter a normalized form. This last step can be in turn decomposed into two: the first one, which generates a set of possible normalization candidates based on the application of certain normalization techniques; and the second one, which selects one of these candidates as the normalized form (in our case, in a score-driven process).

As for the particular normalization techniques employed throughout our system, we decided to try first a combination of two of the traditional approximations to this task (Kobus et al., 2008): the spell checking and the automatic speech recognition metaphors.

2.1 The pipeline

We decided to give our system an *object oriented* approach (using JAVA) as opposed to the *imperative* approach of the original prototype (in PERL). The new system is structured in *processors*, formerly known as *modules* in the prototype, whose goal is to apply a certain process to the input tweets so that we can obtain the normalization candidates of their terms at its output.

The core component of our system is the pipeline, consisting of a classic cascade structure where we can insert an arbitrary number of processors and have their inputs and outputs automatically linked. In this way, the original input of the system becomes the input of the first processor, the output of the first processor is the input of the second one, the output of this second processor is the input of the third one, and so on, until reaching the last processor, whose output becomes the output of the system.

Regarding its design, we have followed good engineering practices and made extensive use of *design patterns*. Among them, it should be noted the use of the *decorator* pattern which, in our context, represents a simple pipeline, allowing us to dynamically stack an arbitrary number of processors. Its combination with the *composition* pattern lets us group them into *stages*, which enable the definition of particular processor sequences while still sharing the same basic processor interface, thus preserving the flexibility of the *decorator*. Thereby, the resulting structure allows for the dynamic construction of different pipeline configurations of varying complexity and different levels of abstraction, not being restricted to the original settings.

The application of the *template* pattern allowed us to factorize great part of the common processes of the components, such as the sequential iteration through all the input tweets, which most of the processors perform. This resulted in a great homogenization of the code, thus simplifying maintenance and allowing us to focus our efforts on the specific implementation of the processing methods in each case.

Moreover, some processors make use of external tools capable of being changed even at runtime — something of special interest in multilingual environments. It should also be possible to integrate them into other external components, so that their logic can be reused by others. All this involves decoupling the processors from the specific implementations of the external components employed, which we have achieved through the use of the *inversion of control* pattern.

Furthermore, communication between the components of the pipeline is done through structured text files, allowing us to gain flexibility as we can integrate and exchange with ease new processing modules regardless of their particular implementation (Vilares et al., 2013). In this case we have used XML along with an implementation of the *abstract factory* pattern for its construction and parsing. This also facilitates possible future migrations to other data representation languages, such as JSON.

Finally, we have created a *dynamic configuration subsystem* based on XML files that allows us to define and instantiate the particular structure of the pipeline on which we want to process the tweets. The advantages of such a subsystem are clear, both for system maintainability and testing:

1. It improves the multilingual support of the system by enabling the definition of configurations that use processors and resources designed for a particular language.

2. It allows for experimentation in a simple, agile and documented (the configuration file itself also serves as documentation) manner.

3. It avoids the necessity of modifying the system source code.

2.2 Configuration before W-NUT 2015

The current processor configuration for Spanish tweet normalization derives from that one used by the initial prototype for its participation in the TweetNorm 2013 task (Alegría et al., 2013). The general procedure works like this: firstly, using processors to prepare the input (preprocessing); secondly, employing those whose purpose is to obtain new normalization forms (candidates generation); thirdly, using those in charge of selecting or filtering the best normalization forms (candidate filtering/selection); and lastly, employing those which prepare the final output of the system (postprocessing). Such setup includes the following processors:

- `FreelingProcessor`, which reads the input data in the TweetNorm 2013 format and uses `Freeling` (Padró and Stanilovsky, 2012) to perform the tokenization, lemmatization and POS tagging (although these tags are not currently in use) of the text of the tweet.

- `MentionProcessor`, `HashtagProcessor`, `URLProcessor` and `SmileyProcessor`, which act as filters for OOVs we do not want to consider for normalization.

- `LaughESProcessor`, which normalizes laugh string representations, as in "ja" for "jajaja".

- `PhoneticProcessor`, which uses a phonetic table to map characters to their phonetic equivalent strings, such as "x" to "por".[1]

- `SMSDictionaryProcessor`, which looks for normalization candidates in an SMS dictionary, for example "también" (too/also) for "tb".

- `AspellProcessor`, which obtains normalization candidates using the spell checker `aspell` (Aspell, 2011), as in "polémica" (controversy) for "polemik". It should be noted that this tool has been customised with a new phonetic table for Spanish, based on the Metaphone algorithm (Philips, 1990) and a new Spanish dictionary extracted from Wikimedia resources.[2]

- `AffixESProcessor`, which identifies and normalizes affix-derived Spanish forms of base words, also supporting phonetical writing, as in the case of "chikiyo" for "chiquillo" (little boy), obtained from "chico" with the suffix "-illo" (little/small).

- `NGramProcessor`, which calculates the scores of those most likely normalization candidates according to the Viterbi algorithm (Manning and Schütze, 1999, Ch. 9) taking as reference the Web 1T 5-gram v1 (Brants and Franz, 2006) Spanish language model.

- `CandidateProcessor`, which selects the top-scoring candidate for each word.

- `ResultProcessor`, which dumps the tweet data obtained by the system to a file using the required format.

2.3 Adaptation for W-NUT 2015

In general, the adaptation process revolved around implementing new processors and integrating new resources to account for the requirements of this new task, such as the use of English instead of Spanish on the new I/O data format, while leaving the base structure of the system untouched. This was precisely the main goal during the refactoring process at the beginning of this project.

The resulting configuration includes the following new processors (see Section 3 for a description of the resources they use):

- `WNUTTweetProcessor`, which parses the structured input (now in JSON format instead of plain text) and obtains the system representation of the tweets.

- `ArkTweetProcessor`, which uses the `ark-tweet-nlp` POS tagger to obtain the morphosyntactic information of the input tweet tokens.

- `WNUTFilterProcessor`, which filters out all those terms that should not be normalized according to the task rules (mentions, hashtags, URLs, etc.) using regular expressions.

[1] The character "x" resembles the multiplication (times) sign ×, which in Spanish is read as "por".

[2] `http://wikimediafoundation.org`

- `LowerCaseProcessor`, which takes all the candidate forms of a token and lowercases them; `AspellCProcessor`, a constrained version of the original `AspellProcessor` described in Section 2.2 (see Section 3 for further details).

- `WNUTNgramProcessor`, which is similar to the previous `NGramProcessor` but with some added modifications to fit the particularities of our new custom language model.

- `WNUTResultProcessor`, which dumps all tweet data generated by the system in the required output format (JSON).

We show in Figure 1 a graphical representation of the architecture of the system both before (left side) and after (right side) the adaptation.

Unfortunately, time limitations prevented us from implementing an English phonetic table for the `PhoneticProcessor`, which would have provided us with mappings such as "two", "too" or "to" for "2". To alleviate this, we did extend the SMS dictionary to cover some of these cases.

It should be noted that because of those limitations we did not address those cases were multiple contiguous tokens of the input tweet should be normalized into a single output token (i.e. the so called "n-1 mappings"). Moreover, since that phenomenon was rare (it appeared in just 11 tweets out of 2950 of the training dataset) we considered that leaving this feature behind would have little impact on the final performance of the system.

3 Integrated resources

The base resources we have used for this task, and on which most of the system processors rely, are the following:

- `aspell` (Aspell, 2011), the well-known spell-checker together with its default English dictionary.

- `ark-tweet-nlp` (Owoputi et al., 2013), a Twitter-focused NLP toolkit from which we have used its POS tagger.

- `BerkeleyLM` (Pauls and Klein, 2011), a Java library and toolset focused on language modeling.

- `Redis`,[3] a noSQL key-value datastore; and the SMS normalization dictionaries, canonical lexicon and training dataset provided by the organizers of the task.

As a result of processing the previous resources, we have obtained the following additional ones:

- A global SMS normalization dictionary implemented as a `Redis` datastore, whose entries were extracted from the two normalization dictionaries and the training dataset provided by the organizers.

- A Kneser-Ney language model (Kneser and Ney, 1995) of the target domain (standard tweet text) obtained with the `BerkeleyLM` tools taking as input tweets of the training dataset.

- A new English dictionary for `aspell` built on the canonical lexicon.

With respect to the differences existing between the configurations of the system for constrained and unconstrained runs, there is only one. In the case of the *constrained* run, since only *off-the-shelf* tools are permitted, the `aspell` spell-checker was employed using its default dictionary but filtering its retrieved candidate corrections taking as reference the canonical lexicon; i.e. only those candidates that could be found on this lexicon were taken into account. On the other hand, in the case of the *unconstrained* run, `aspell` was used instead with the dictionary obtained from the canonical lexicon. The rest of the processors and their parameters remained the same.

Moreover, although we also considered the use of the Web 1T 5-gram v1 language model in the unconstrained run, our preliminary tests showed that the results obtained were very poor in this case, as we further comment in Section 4.

4 Evaluation

Table 1 shows the results obtained for the *training* corpus. It should be noted that these correspond to a slightly *overfitted* system, since we inadvertently used a language model built using the whole training dataset (for candidate selection) in our 10-fold cross-validation framework. Nevertheless, this also gave us an interesting clue to the main performance bottleneck of our system, as we will discuss below.

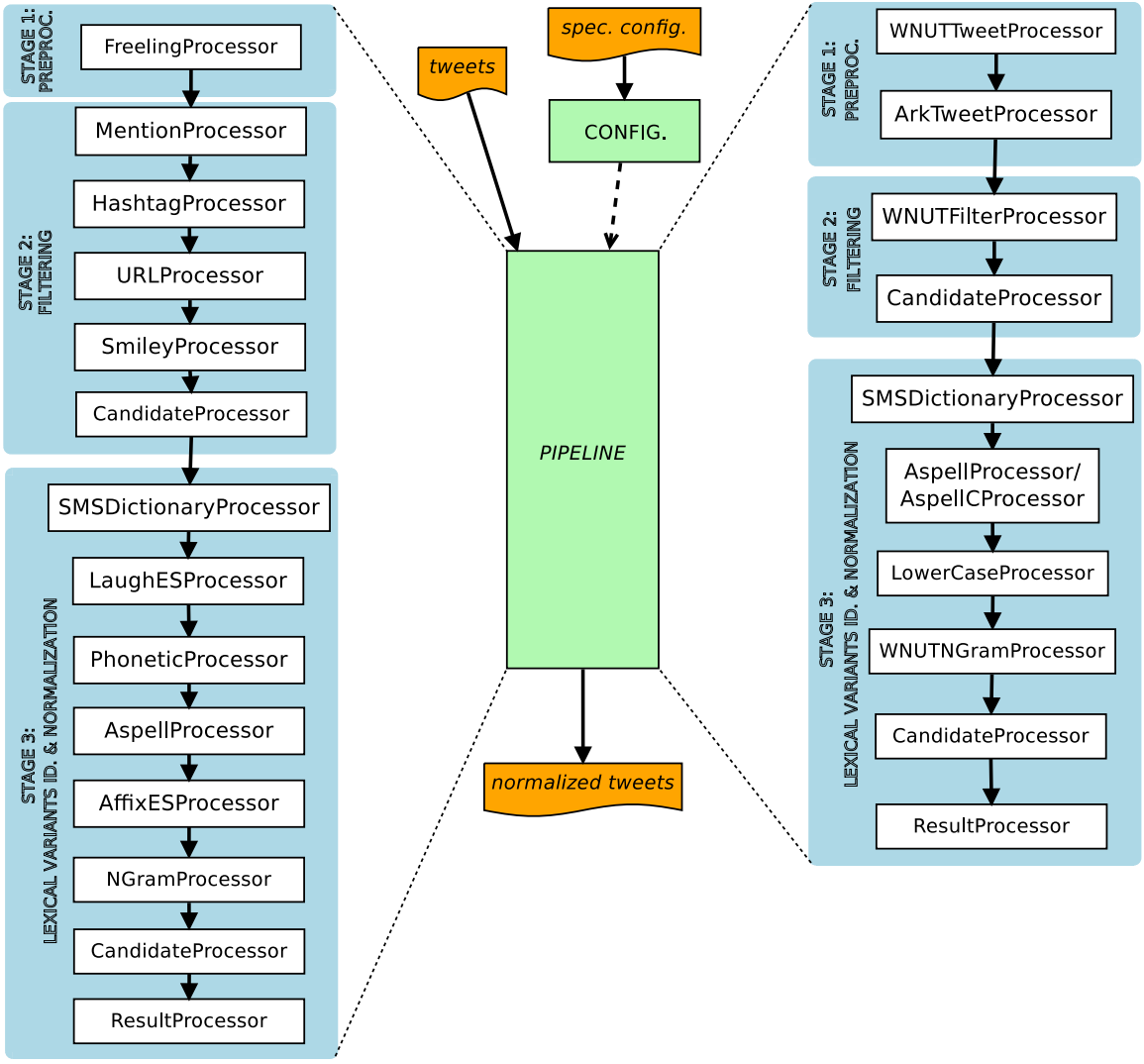

Figure 1: Original pipeline (left) and pipeline adapted for W-NUT 2015 (right) integrated into the architecture of the system.

	precision	**recall**	**F1**
constrained	0.8956	0.8746	0.8850
unconstrained	0.8914	0.8739	0.8825

Table 1: Training results.

	precision	**recall**	**F1**
constrained	0.4646	0.6281	0.5341
unconstrained	0.4592	0.6296	0.5310

Table 2: Testing results.

Table 2 shows the results obtained for the *test* corpus. At the sight of these figures, which differ considerably from the previous ones, we decided to analyse them in more detail. For this purpose, we obtained a recall metric on the scope of the candidates proposed by the system; in other words, we wanted to see how many times the correct candidate corresponding to a token of the dataset was among the ones considered by the system. The resulting ratio came to `0.87`, which means that most of the times we had had the chance to select the correct normalization form for a given non-standard token but the system failed to make the selection, and is also a consistent figure with respect to those shown on Table 1. This was not a big surprise for us, mainly because it is a well-known problem we have been aware of since we started working on (Spanish) tweet normalization. Therefore, we can conclude that the performance bottleneck of our system is still the candidate selection process, which is heavily influenced by the

[3] `http://redis.io/`

language model in use.

In this respect, tuning experiments were also made by extending our unconstrained configuration through the addition of the Web 1T 5-gram v1 English language model as a knowledge source. Only unigrams and bigrams could be used because of unsolved memory limitations. However, in contrast with previous experiments performed for Spanish, the resulting performance was unsatisfactory. Because of this, the use of these language models for our final submission was dismissed. According to our analysis, the cause for this seems to be the great differences, at both the lexical and syntactical levels, between the texts used to build this model, which could be considered as "regular" texts, and those corresponding to tweets, which agrees with the observations of Chrupała (2014). As illustrative examples of this type of expressions we can take "I like them girls" and "Why you no do that?", which are lexically correct but not syntactically valid, so language models built using regular texts will not recognize them. In the case of our previous experiments on Spanish, this difference was not so clear.

5 Conclusions and Future work

We have presented in this work the tweet normalization system used by our group to participate in the W-NUT 2015 Normalization Task which, in turn, is an adaptation of another existing Spanish tweet normalization system.

Within the scope of this task, it became clear that most of the normalization mistakes made by our system occurred during the candidate selection stage, as it was unable to determine the correct normalization term obtained in previous stages from the set of candidates available. The reason for it is that we do not have at this very moment enough training data to build a representative language model of the target domain (normalized text of English tweets).

Furthermore, there is another type of normalization phenomena which, at this moment, cannot be correctly handled by our system: n-1 mappings. This is due to the initial approach we took for this system, which only considered 1-1 and 1-n mappings, but not n-1 mappings, together with our time limitations.

All that being said, as future lines of work we are considering the following improvements to our system:

- Obtaining a representative language model of the target domain by using a larger normalized tweet corpus. This corpus will be comprised of tweets without non-standard words, so we can still capture the morphosyntactic structure of these texts (Yang and Eisenstein, 2013).
- Using POS tags and syntactic information to improve the candidate selection process.
- Integrating a classifier in the extraction process of the final normalization candidates, taking as features aspects such as the syntactic and morphosyntactic information obtained, their probability according to the language model, whether they were selected or not by the Viterbi algorithm, their string and phonetic differences with respect to the original form, etc.
- Keeping the canonical lexicon updated using resources like Wikipedia, since the language model construction process relies heavily upon a good lexical reference in order to correctly discard non-standard words.

Moreover, we intend to study the application of tweet normalization, for both Spanish and English tweets, in *opinion mining* tasks (Vilares et al., 2015).

Acknowledgments

This research has been partially funded by the Spanish Ministry of Economy and Competitiveness and FEDER (through project FFI2014-51978-C2-2-R) and by the Autonomous Government of Galicia (through grant R2014/034).

References

Iñaki Alegría, Nora Aranberri, Víctor Fresno, Pablo Gamallo, Lluís Padró, Iñaki San Vicente, Jordi Turmo, and Arkaitz Zubiaga. 2013. Introducción a la tarea compartida Tweet-Norm 2013: Normalización léxica de tuits en español. In *Tweet-Norm 2013. Tweet Normalization Workshop 2013*, volume 1086 of *CEUR Workshop Proceedings*, pages 1–9. CEUR-WS.org.

GNU ASPELL (rel. 0.60). 2011. Available at: `http://aspell.net` (visited on May 2015).

Timothy Baldwin, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on

noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text (WNUT 2015)*, Beijing, China.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram Version 1 (ref. LDC2006T13). DVD. Distributed by Linguistic Data Consortium.

Grzegorz Chrupała. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–686. ACL.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: makn sens a #twitter. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011) - Volume 1*, pages 368–378. ACL.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):5:1–5:27.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-95)*, volume 1, pages 181–184. IEEE.

Catherine Kobus, François Yvon, and Géraldine Damnati. 2008. Normalizing SMS: Are Two Metaphors Better Than One? In *Proc. of the 22nd International Conference on Computational Linguistics (COLING'08) - Volume 1*, pages 441–448. ACL.

Paula López Rúa. 2007. Teaching L2 vocabulary through SMS language: Some didactic guidelines. *Estudios de lingüística inglesa aplicada*, 7:165–188.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge (Massachusetts) and London (England).

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proc. of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 380–390. ACL. Toolkit available at: `http://www.ark.cs.cmu.edu/TweetNLP/` (visited on May 2015).

Lluís Padró and Evgeny Stanilovsky. 2012. Freeling 3.0: Towards Wider Multilinguality. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA). Toolkit available at: `http://nlp.lsi.upc.edu/freeling/` (visited on May 2015).

Adam Pauls and Dan Klein. 2011. Faster and Smaller N-gram Language Models. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011) - Volume 1*, pages 258–267. ACL. BerkeleyLM source code available at `https://code.google.com/p/berkeleylm/` (visited on May 2015).

Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language*, 7(12):39–43.

Jesús Vilares, Miguel A. Alonso, and David Vilares. 2013. Prototipado rápido de un sistema de normalización de tuits: Una aproximación léxica. In *Tweet-Norm 2013. Tweet Normalization Workshop 2013*, volume 1086 of *CEUR Workshop Proceedings*, pages 39–43. CEUR-WS.org.

David Vilares, Miguel A. Alonso, and Carlos Gómez-Rodríguez. 2015. On the usefulness of lexical and syntactic processing in polarity classification of Twitter messages. Accepted for publication in *Journal of the Association for Information Science and Technology (JASIST)*. DOI 10.1002/asi.23284.

Yi Yang and Jacob Eisenstein. 2013. A Log-Linear Model for Unsupervised Text Normalization. In *Proc. of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 61–72. ACL.