Robust Morphologic Analyzer for Highly Inflected Languages

Andrés Tomás Hohendahl

Laboratorio de Estereología y Mecánica Inteligente. Dto. de Ing. Mecánica
Facultad de Ingeniería, Universidad de Buenos Aires, Buenos Aires, Argentina
Instituto de Ingeniería Biomédica, Facultad de Ingeniería Universidad de Buenos Aires
Buenos Aires, Argentina

José Francisco Zelasco, Judith Donayo

Laboratorio de Estereología y Mecánica Inteligente. Dto. de Ing. Mecánica
Facultad de Ingeniería, Universidad de Buenos Aires, Buenos Aires, Argentina

Abstract. We present a multilingual robust morphologic tagger and tokenizer for highly inflected languages like Spa-nish, with efficient spell correction and 'sound-like' word inference, obtaining some semantic extraction even on parasynthetic and unknown words. This algorithm combines rules, statistical best-affix-fit along with a language estimator. A rich flag set controls the internal behaviour. The system has been designed for efficiency and low memory footprint, using data structures based on simple available affixing rules. Our system, packed with a Spanish dictionary of 83k lemmas and 5k rules, recognizes 2.2M exact words, the guessing word-space is many times this much.

1 Introduction

In any system, to achieve a good quality human-machine communication schema, the first processing stage is critical; turning out to be rather complex when dealing with free text, including possible mis-spells and Out Of Vocabulary words (OOV).

To achieve good NLP processing, a robust input stage is needed. Even best statistical POS taggers, needs morphologic dictionaries, which itself tur-ned out to be 'the real problem' we address here.

Another challenge was to design a system to run on limited resources (PC, mobile, etc.). Linguistic information is also sparse and difficult to collect to fit into efficient datasets. We adopted open-source dictionaries & data [15]. By digging into this data-sets we found a way to build grammatical and semantic information based on the morphologic word derivation processes used in spell correction.

The main contribution of this work is to build a robust, compact linguistic tagger, capable of dealing with the aforementioned 'real world problems', running efficiently on limited resources; being capable of splitting 'dirty' text sentences doing a best-effort morphologic analysis on each tag with grammatical and semantic extraction,

TomÃąs Hohendahl A., Zelasco J. and Donayo J.
Robust Morphologic Analyzer for Highly Inflected Languages.
DOI: 10.5220/0003015301120118
In *Proceedings of the 7th International Workshop on Natural Language Processing and Cognitive Science (ICEIS 2010)*, page
ISBN: 978-989-8425-13-3

correcting spell errors and providing a statistic quality measure for all correc-tions and OOV. words.

The first hard issue was to address was the huge amount of data to be stored [2], along with the ambiguity [3], present at every level of analysis [16].

Unlike English, Spanish has quite a large amount of combinable derivations & inflections [7]; with circa 1k prefixes [6]; yielding a word space of many millions. The database must contain semantic, gram-matical, statistic & some ontological data, similar to data found on WordNet [3] [19].

We created a robust error handling algorithm by properly tweaking known data structures (Trie & TST) [13][14] into fast reversible affix algorithms. The final system is capable of handling regular and irregular affixation clogged with spell-errors.

Many rules and formats used by this algorithm were originally created only for spell checking [8].

We included a new Spanish phonetic distance measure [19], implementing also Metaphone and Soundex. We included a language estimator improving our former work [1].

2 Split & Classify

'Where to split a string' is by no doubt, a hard issue. To achieve this, we ported & enhanced JLex, obtaining a C# attributed lexer capable of recognize digits, sci-numbers, romanic, hex, phones, emails, url, uri, math symbols, punctuations, times & dates, times, etc; sending any 'suspected' words into the next stage: the morphologic analyzer.

The next main challenge was to include a huge inflected-word dictionary. Many known squeezing techniques exists [4], the goal is to find a good balance between size and retrieval performance.

We tried out many methods [17] [18], choosing to tore all data in specially tweaked TRIE [13] and TST [14]; obtaining a fast & compact system.

2.1 Morphologic Rules

The affix-flexion rules were based on a format called AFFIX [8], augmented to allow grammar & semantic recall. ISPELL [11] format was used to express the synthesis of flexion rules, including the changes of the root word. This new created notation, can describe rather complex transformation patterns, combining regular-expression pattern syntax [9].

The de-flextion algorithm is controlled by many flags allowing us to infer OOV. & parasynthetic words, by stripping out the affixes on both ends; obtaining semantics tags during the strip; being especially useful to recognize scientific words not present in dictionaries like DRAE [10] or others.

3 Dictionary Format

By adopting the well-known ISPELL [11] format we created a clear way to write down the rules. Our dictionary consists of a first rule-section in ASPELL [8] format, encoding flags to control the affix processing. Each rule is a set of entries describing specific affix operations, with some extra info. Following there is a tagged-root-word section, where each lemma is annotated with each and every applicable rule name, delimited by '**/**'. Example:

amar/VXYLHp/VMN/3M2R

The third field '**VMN**' uses Eagles2 tags. The next field: '**3M2R**' encodes semantics & grammar info.

Let's take a look at the "**V**" rule after the first slash, shown in ISPELL format:

flag *V:
#GP present, 1º person, singular
A R > -AR,O # am**AR** > am**O**

The condition indicates the root word should end with '**AR**', then '**-AR**' states: we must remove those two letters, and then finally append '**O**' to get the inflected word transforming: am**AR** into am**O**.

3.1 Additional Functions

Some format enhancements have been added to simplify complex feature expression, allowing us to express individual rule-entries as combinations.

#GR common noun | adjective

Comment-embedded meta-commands like '**#GR**' allow us to build accumulative information. Those functions are used during import-time of ISPELL rule-data, being converted into a compact AFFIX-sub-format for the final internal 'es-ES.dic' file.

4 The Morphologic Analyzer

One logical way to find the inflection root of an unknown word is to un-apply each and every known inflection rule combination and see if the remains coincide with a root word in our dictionary, and then check if this word admits the used rules. This clearly turns out to be a combinatory-NP problem.

A simpler strip-guess mechanism, used by many popular packages [8] [11] [12] is explained: First we start stripping off the suffixes by reversing the applicable rules, each success adds to a list of prefix-strip candidates; then for each prefix-strip candidate, we see whether if it is a root, if not we try to rip off any prefix, leaving a clean root candidate word to be found on our root-dictionary. The time spent to do this is quite a lot, requiring many de-affixation operations and a lot of dictionary lookups.

For example if we have the words on a *P* factor balanced-tree, it gives a complexity in the order of:

$$O(Se*Pe*Nw*Log_P(Nw))$$

At least storing all word in an all in-memory hashtable the complexity can be lowered only to:

O(*Se*Pe*1)***

Where (Se~4500: Suffix & Pe~330: Prefix entry counts) for our Spanish system. This still results in over 1.1 million de-affixations and seeks operations per word, which yields a very slow system.

Our system changes the number of lookups by using TRIES and TST on the prefixes against the unknown word, obtaining a linear performance, based only on how-many characters in the word coincide in-order with the last (suffix) or first (prefix) patterns. The lookups and de-affixations were 10 to 50, independent on the root dictionary size, as we used a fast hash-type like a TST for root lookup. Our analyzer was an average of 10^4 times faster than popular strip-guess system.

After the tokenization and lemmatization, a final stage detects and merges locutions, abbreviations, 'spelled numbers'; strips enclitic pronouns from verbs and detects basic named entities.

5 Internal Structure

Our special tree structure called Trie [13] [17], to store the transformed affixes (suffix / prefix) holding multiple elements by node. This allows the rules to be selected directly in as many comparisons as letters has the affix; this reduces 4500 comparisons [12] to an average of 3.2, for that rule (x 1400:1). We use an internal Trie for rules to improve speed, testing first the longest found affix.

When the rule holds nested structures, a unified Trie is used which keeps all the combinations optimally, with a mean complexity of: $\boldsymbol{O(\log_2(k + n))}$. (Mehlhorn) [18] and [17], being "***k***" the number of rules and "***n***" the affix letter count on each rule, ranging from 1-3 letters (in average: 3 comparisons).

For the storage and search of root words, we also decided to use a variant of a custom TST [14], performing similar to a hash table [17], allowing us to find words (while seeking) with N character or using even wildcards. This speeds up and simplifies the work to find a misspelled word, while testing a possibly misspelled one.

We implemented variants of both TST and TRIES that allows finding approximate roots to address avoiding the diacritics; being able of generating useful alternatives in case of missing or wrong letters. They exhibit at the most **O(**N*P**)** additional comparisons; (where N is the number of letters of the word and P is the number of "allowed errors" (nº of diacritics) whose average is between 2 and 4).

5.1 Data Formats

We allow a cumulative effect in the affix-rule format used to obtain the labelling of words in order to avoid numerous repeated labels and also to group words under a same head-label. This is very useful at the time of adding words and publishing the list of the similar ones. The extended format is very simple to understand and maintains full compatibility with the previous ASPELL dictionary format (being able to read older formats); we only add 2 special start-line-codes: "*" corresponding to

the grammar labels and "%" for the rules, with minimal overhead.

The used/proposed affix compression mechanism is very compact and fast for reading and decoding the dictionary files (<0.3seg./500k), being many times better than the standard Open-Office (OO) [15] file format. Also due to the file format enrichment, it has a more expressive capability.

Spanish "es-ES.dic" [15] is 712kb with 48k root-words whereas ours is 850k, with 81k root-words and 526 combinable tag categories; both have the same aprox. ~4500 morphologic rule-entries.

5.2 Error Correction

When a word is not recognized, a spelling algorithm is launched, making changes to guess a misspelled word, using a letter permutation/change technique sometimes called "the poor man speller" trying to guess a typical misspell. If we got success, the word is tagged as "unknown" and the found alternatives are added, each with a similarity [19] measure.

6 Performance & Testing

The dictionary processed 80k words/second, on a 2660 word-corpus (Ortega&Gasset) using ~18Mb memory under .NET. The general score of error detection was 89.64% and the successful lexical recognition was 38%. We built a full dictionary including definitions of DRAE [10].

We benchmarked on a Spanish Blog of 57780 words: 73% were tagged at 2366 words/second, 7.5k misspells were guessed at 15.4 words/s rate. Final Recall was 99.6%. 0.4% non-words (dirt) remained untagged. System: (XP+SP3+i386/E5300/2Gb)

7 Conclusions

The actual system has successfully fulfilled the design goals, being fast, lightweight and having the capacity to handle different levels of precision and a wide range from exact to approximated word recognition; offering a set of "similar" and "corrected" phonetically sorted words, useful for later NLP processing. Our libraries and systems are built in C# using .NET to be platform independent.

Trying to compare our system to others, we could not find any benchmarks, spell-error handling or phonetic enabled morphologic analyzers out there. To mention a few: AGME use a 19Mb data file. Freeling [5] use ~60Mb RAM for a 600k words. No information was found for many other analyzers: [6] [7]. The GATE framework is a good NLP tool, but we found no support for Spanish, errors and morphologic analysis.

7.1 Perspective & Future Work

One of the today's most challenging issues are the correct handling of Free Natural Language. The presented system adds some power to this crusade being able to handle many errors and decently guess words in a 'dirty' & 'misspelled' environment.

The actual target is Human Computer Interface (HCI) for Dialog/Chat/Email/Blog text processing.

We developed a dictionary editor to handle rules and semantic tags. With this tool was able to add over 300 of Greek, Latin, and German affixes, including a lot of 'human-like' semantic data.

An interesting field of applications is recognition of parasynthetic words in bio/medical records and scientific text. Medical dictionaries Espasa [21], has only 23k words, while Snomed-CT [20], has >600k mostly OOV multi word-terms.

We think that the semantic extraction based on morphology may prove useful for further NLP processing like Word Sense Disambiguation (WSD).

Indeed any human-machine dialog systems may benefit on a fast, robust and compact tagger like this.

References

1 Hohendahl, Andrés T. & Zelasco, José F. 2006. Algoritmos eficientes para detección temprana de errores y clasificación idiomática para uso en procesamiento de lenguaje natural y texto, WICC2006 - ISBN 950-9474-35-5

2 Diccionarios españoles: http://www3.unileon.es/dp/dfh/jmr/dicci/012.htm

3 Pedro Luis Díez Orzas 1999. Estudios de Lingüística Española LA RELACIÓN DE MERONIMIA EN LOS SUSTANTIVOS DEL LÉXICO ESPAÑOL: CONTRIBUCIÓN A LA SEMÁNTICA COMPUTACIONAL Volumen 2 (1999) ISSN: 1139-8736

4 Shannon, Huffman compression: http://www.cbloom.com/algs/statisti.html

5 FreeLing: http://www.lsi.upc.es/~nlp/freeling/

6 FLANOM: Flexionador y lematizador automático de formas nominales. Santana, O.; Pérez, J.; Carreras, F.; Duque, J.; Hernández, Z.; Rodríguez, G. Lingüística Española Actual XXI, 2, 1999. Ed. Arco/Libros, S.L. 253/297

7 FLAVER: Flexionador y lematizador automático de formas verbales. Santana, O.; Pérez, J.; Hernández, Z.; Carreras, F.; Rodríguez, G. Lingüística Española Actual XIX, 2, 1997. Ed. Arco/Libros, S.L. 229/282

8 ASPELL Affix compression: http://aspell.sourceforge.net/man-html/Affix-Compression.html

9 Expresiones Regulares: http://www.regular-expressions.info/

10 DRAE Diccionario de la Real Academia Española http://buscon.rae.es/diccionario/drae.htm

11 ISPELL www.gnu.org/software/ispell/ispell.html

12 NetSpell http://sourceforge.net/projects/netspell/

13 TRIE http://www.cs.bu.edu/teaching/c/tree/trie/

14 TST -Ternary Search Tree: www.nist.gov/dads/HTML/ternarySearchTree.html

15 Open Office Dictionaries: http://lingucomponent.openoffice.org/spell_dic.html

16 Relaciones morfoléxicas prefijales del español. Santana, O.; Carreras, F.; Pérez, J.; Rodríguez, G. Boletín de Lingüística, Vol. 22. ISSN: 0798-9709. Jul/Dic, 2004. 79/123.

17 J Bentley & R - Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, 1997

18 Mehlhorn, K. Dynamic Binary Search. SIAM Journal on Computing 8, 2 (May 1979), 175-198.

19 Hohendahl, A.T.; Zanutto, B. S.; Wainselboim, A. J.; “Desarrollo de un algoritmo para la medición del grado de similitud fonológica entre formas escritas” SLAN2007. X Congreso Latinoamericano de Neuropsicología 2007, Buenos Aires, Argentina
20 SNOMED CT (Systematized Nomenclature of Medicine--Clinical Terms) http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.htm
21 Editorial ESPASA CALPE http://www.espasa.com