PANGLYZER: SPANISH LANGUAGE ANALYSIS SYSTEM

David Farwell Steve Helmreich Wanying Jin Mark Casper
Jim Hargrave Hugo Molina-Salgado Fuliang Weng

Computing Research Laboratory
New Mexico State University

ABSTRACT

The purpose of this paper is to describe the functions and procedures of the eight components of the Panglyzer Spanish analyzer of the knowledge-based engine of the Pangloss machine translation system. The Panglyzer follows a multi-pass approach consisting of preprocessing, part-of-speech tagging, phrase recognition, proper name classification, phrase analysis, clause recognition, clause analysis and reading ranking.

Introduction

The function of the Spanish analysis component of the PANGLOSS system, or PANGLYZER, is to provide for each clause in the input text a set of possible meaning representations ranked on the basis of likelihood. These are then handed to an augmentor which either interactively or automatically selects a representation from among the set of possibilities and fills in various sorts of information to produce a reading that is compatible with the context and that can act as a basis for generation.

The approach has been to develop the system in a bottom up manner focusing on providing layer after layer of increasingly abstract analysis in a multi-pass process. This multi-pass architecture allows for semi-independent module construction, incremental development, and is amenable to robust performance. Each level of analysis is based on a focussed type of knowledge and, to the extent possible, exploits proven techniques. The levels alternate between recognition (segmentation) modules and analysis modules. For example, the preprocessor segments text into words while the part-of-speech tagger analyzes their part of speech. Since a high premium has been placed on robustness, we are following an iterative approach to design which relies on rapidly producing an initial prototype and then following a short test and revise cycle. Thus, at this point, all but the deepest level of analysis produces throughput and the on-going objective is to improve the accuracy of that throughput from one test cycle to the next

System Architecture

The architecture of the PANGLYZER is a by-product of the bottom up approach to development described above. It consists of eight components that sequentially process the data. The first is a Preprocessor which converts a text length input ASCII character string into a file of PROLOG data structures and then builds PROLOG strings corresponding to the sentences of the input text. The second is a Spanish Part-of-Speech Tagger which provides a standard data structure for each item of an input

PROLOG string which includes a part of speech, relevant inflectional information, and position index. The third component is the Phrase Recognizer which groups contiguous elements of the input part-of-speech tagged string into phrase length chunks, inserting brackets around the chunks and assigning phrasal categories. The fourth component is a Proper-Noun Classifier which, operating over the entire text, assigns a semantic category such as personal name, place name, company name, and so on, along with relevant inflectional information to each element of the input tagged as a proper noun. The fifth component, the Phrase Analyzer, provides a set of possible semantic representations for each phrase of the input string. The sixth component is a Clause Recognizer which groups contiguous sets of phrase analyses in the input string into clause length chunks, inserting labeled brackets indicating clause types around the chunk and indexing its positions within the clause. The seventh component, the Clause Analyzer, assigns syntactic dependency relations such as head, subject-of, object-of, circumstantial-modifier-of, and so on to the various constituent phrases of each clause. The eighth component is the Reading Ranker which provides a likelihood score for each possible combination of phrase analyses given their contexts within the clause.

1. Preprocessor

The function of the Preprocessor is to convert a Spanish input text, in the form of an ASCII file, into a file of data structures. First, the input ASCII file is converted into a file of PROLOG strings each of which contains a single atom corresponding to a word or punctuation mark in the input text. In regard to the example text,

> *Al momento de su venta a Iberia, VIASA contaba con ocho aviones, que tenían en promedio 13 años de vuelo.* (At the time of its sale to Iberia, VIASA had eight airplanes, which had an average of 13 years of flight time.)

the initial conversion process results in a file of PROLOG strings of the form:

['Al'].[momento].[de].[su].[venta].[a].['Iberia'].[','].['VIASA']....

In the second step, these unit strings are concatenated into PROLOG strings corresponding to the sentences of the input text. The Preprocessor takes as input the file of unit PROLOG strings above and yields a file of Prolog strings of the form:

['Al',nomento,de,su,venta,a,'Iberia',',','VIASA',...].

2. Spanish Part-of-Speech Tagger

The Spanish part-of-speech tagger automatically labels words with part-of-speech categories. The tagger uses an on-line dictionary, Spanish analytical morphology and fix-up rules to tag words. Fix-up rules use local context to alternate the part-of speech in case the part-of speech is inappropriate.

The morphological analyzer takes as input a sentence, analyzes each word and generates the corresponding lemma. It also produces morpho-syntactic information gleaned from the word form. Tag category lookup order was experimentally established and is used to decide the most likely analysis in case of ambiguity. The morphological analyzer uses the Collins Spanish-English Dictionary for single word lexical items and a custom built database for phrasals. The morphological analyzer supports all verbs found in Collins and their inflected forms, as well as most inflectional morphology for nouns, adjectives, pronouns and articles.

The preliminary tagging by the morphological analyzer results in a string of the form:[1]

[1] Although *al* is tagged as a preposition, under analysis it is treated as the preposition *a* and the article *el.* See the analysis at the end of Section 5.

```
[['Al'/al/preposition],
 [momento/momento/noun(masculine,sg)],
 [de/de/preposition],
 [su/su/adjective(neuter,sg)],
 [venta/ventar/verb(pres_ind,sg_l)],
 [a/a/preposition],
 ['Iberia'/[]/proper_noun(unknown)],
 [',','/',',punc(comma)],
 ['VIASA'/[]/proper_noun(unknown)],...
```

The fix-up rule component takes as input the list of word/lemma/tag sublists in the output above and uses local syntactic cues to repair common mistakes. The fix-up rules themselves try to match sequences of words, lemmas, parts of speech or inflectional information against the input. If a match is found, it triggers the revision of some particular part-of-speech tag. For instance, in the example above, the verb tag for *venta* is altered to a noun by a fix-up rule that changes the tag of a verb immediately following a possessive adjective to a noun (if the word in question has a possible noun reading).

3. Phrase Recognizer

The Phrase Recognizer identifies basic grammatical constructions at the phrase level using a Definite Clause Grammar (Pereira & Warren, 1980; Huang, 1988).[2] The DCG used by this module is composed of a collection of rules that identify noun phrases np, verbal constructions vc, prepositional phrases pp, proper noun phrases pn and preposition plus proper noun phrases ppn. Some words fall into special non-phrasal single-word categories such as conjunctions cj, complementizers cm and punctuation pc. The residue of unassimilated words are simply tagged as single-words sw.

Examples of rules used in the DCG to tackle verb constructions and noun phrases are:

```
verb complex:     vc → adverb, verb.
                  vc → auxiliary, verb.

noun phrase: np → article, noun.
             np → article, npl.
             npl → noun, adjective.
```

It should be noted that these are generally not complex. Except for certain cases where the semantic analysis of the whole is assured to be correct such as phrases like *30 millones de dólares* (30 million dollars), each prepositional phrase, appositive, and complement is analyzed separately. (See footnote 2).

The input file used by the Phase Recognizer is the output from the Tagger. The Phrase Recognizer takes the first words in the input sentence and tries to match them with one of the constructions determined by the DCG. When one of the rules succeeds, the sequence of words which satisfies that rule is bracketed and labeled for the specified category. For instance, for the example sentence, the output begins with two prepositional phrase spanning elements 1 through 5.

[2] These grammatical constructions do not directly correspond to any theoretical linguistic constructs. Instead, they are intended to identify potential arguments, predicates or adjuncts.

```
[[pp,[1,2],[[[1,1],['Al'/al/preposition]],
           [[2,2],[momento/momento/noun(masculine,singular)]]]]
[[pp,[3,5],[[3,3],[de/de/preposition]],
           [[4,4],[su/su/adjective(neuter,sg)]],
           [[5,5],[venta/venta/noun(feminine,singular)]]]],...]
```

The recognition of grammatical structures proceeds sequentially until all the words in the input have been given a phrasal analysis. The final output file consists of lists of sublists corresponding to the basic phrasal constructions determined by the DCG. Each list corresponds to a sentence of the input text.

4. Proper-noun Classifier

The Proper-noun Classifier provides each proper noun with a unique classification from the following list of categories: *government entity, geographical entity, corporate entity, human name, date, professional title.* If classification into one of the above categories fails, a default classification of *other* is given. The component has a multi-pass architecture. That is, all proper nouns in a text are sent through pass one. Those that fail to be positively identified are sent through pass two. Again, those failing to be positively identified undergo pass three. On the first pass, classification is attempted via a list-based matching scheme. On the second pass, contextual information is used to further disambiguate non-unique tags from the first pass. Finally, default rules are applied to insure a unique classification for each proper noun.

The first pass involves searching lists for a complete or partial match with the proper noun under consideration. The lists used were compiled from a variety of resources including gazetteers, phone books, etc.

Pass one classification of a given proper noun actually is carried out in two phases. First, an attempt is made to uniquely tag the proper noun as a date, as a corporate entity, or as a title. If this process succeeds, the proper noun is considered to be positively identified, the tag is assigned and processing ends for that item. Once a tag has been assigned, the proper noun and tag are stored for future reference. If it fails, then the proper noun is exhaustively tagged with every possible classification found by matching the noun against the word lists. In doing so, a three-level tag-ranking system is employed to aid in future disambiguation. If this exhaustive tagging procedure results in only one possible tag, then the item is considered to be positively identified and the tag is assigned.

Pass two attempts to select one of the multiple tags of the exhaustively tagged proper nouns from the first pass. The first context considered is the set of already uniquely classified proper nouns. Any partial overlap with such nouns are given the corresponding classification, provided certain partial match criteria are satisfied.

The second type of context-based selection employed in pass two involves an analysis of the surrounding words, their parts of speech, and their proper-noun classifications (if any). A number of experimentally developed heuristics, involving different combinations of one or more of the above types of information, are applied to completely select a tag for the proper noun, or, at least, to further reduce the possible tags by filtering out one or more. Mechanically, the rules are applied starting from those which utilize the least amount of contextual information to those which utilize the most contextual information. If either of these processes select a single tag, the item is considered to be positively identified and that tag is assigned.

The final pass through the text involves applying default rules to any remaining multiply classified proper nouns. If exactly one of the possible classifications has highest rank, then it is the classification assigned to the proper noun; otherwise assign the default tag of *other.*

5. Phrase Analyzer

The function of the Phrase Analyzer is to assign to each phrase identified by the Phrase Recognizer, as extended by the Proper-Noun Classifier, a set of possible meaning representations. The Phrase Analyzer constructs all possible interpretations for each phrase and then passes them on to the Clause Analyzer, which is to compose the semantically coherent readings at the clause level.

The performance of the Phrase Analyzer depends heavily on lexical information stored in the PANGLYZER's two lexicons: the Spanish lexicon, which encodes information specific to Spanish, and the word sense lexicon, which contains the semantic information which Spanish shares with other languages in general. Items in the two lexicons are tied to together by word sense tokens. These tokens are drawn from the sense definitions of Longman's Dictionary of Contemporary English (LDOCE) (Proctor, et al., 1978).

Below is an example of an entry for a singular form of a Spanish noun in Spanish lexicon. This entry provides information about agreement, gender, syntactic case, along with a word sense token indicating the item in the word sense lexicon with which it is associated.

```
se_form(momento,ts,m,_F,moment_0_l).
```

Entries in the word sense lexicon of the conceptual category entity provide information about semantic class, countability, LDOCE semantic class, and LDOCE subject domain as shown in the following entry.

```
ent(moment_0_l,nrm,time,c,abstract,open).
```

Input to the Phrase Analyzer comes from the Phrase Recognizer, as augmented by the Proper-noun Classifier. In regard to the example sentence, the input begins with:

```
[pp,[l,2],[[l,l],['Al'/al/preposition]],
        [[2,2],[momento/momento/noun(masculine,singular)]]]
```

which corresponds to Spanish prepositional phrase *al momento.* The Phrase Analyzer uses a Spanish DCG to parse *al momento.* These Spanish grammar rules are compatible with the grammar rules in the Phrase Recognizer. The syntactic information on phrase category supplied by the Phrase Recognizer (i.e., np, pn, pp, ppn, etc.) is used to index the corresponding DCG rules of the Phrase Analyzer.

The grammar rules in the Phrase Analyzer also access the lexicon entries and unify the syntactic and semantic information in the entries to produce the meaning representations like the one shown below. For the example phrase, the Phrase Analyzer produced two possible analyses:

```
[[mod,[string,al,momento],
    [g_rel,B],[k_rel,t_loc],[t_rel,C],
    [ent,[string,el,momento],
        [type,nrm],[class,time],[agree,ts],[det,spin],[quant,unq],
        [e_desc,[string,momento],
        moment_0_l]]],
[mod,[string,al,momenta],
    [g_rel,E],[k_rel,condition],[t_rel,F],
    [ent,[string,el,momenta],
        [type,nrm], [class,force],[agree,tu],[det,spin],[quant,unq],
        [e_desc,[string,momento],
        momentum_0_2]]]]]
```

6. Clause Recognizer

The function of the clause recognizer is to group the phrases into clauses, inserting labeled clause boundaries in the process. It does this by applying a DCG for sequences of phrasal categories in top-down, depth-first, left-to-right fashion. It recognizes several types of clauses finite, relative, participial, infinitival etc. as well as groups of phrases that do not correspond to any of the sequences expected by the DCG. These are assigned to a no-clause category.

Generally, the DCG recognizes a sentence length input sequence of phrases as a single finite clause, as a single finite clause followed by other clauses or as a single non-clause. The grammar defines finite clauses as consisting of zero or more phrases followed by a finite verbal construction which is followed by zero or more other phrases. A phrase may be a relative clause, a passive participial clause, an infinitival clause, or some basic phrase. A relative clause consists of a relativizer followed by a finite verbal construction possibly followed by some number of phrases. A participial clause consists of a participial possibly followed by some number of phrases. Finally, an infinitival clause consists of a preposition followed by an infinitive possibly followed by some number of phrases. The resultant output, in simplified form, for the example sentence is:

```
[[finite,
  [[1,1],[pp,['Al',momento]]],
  [[1,2],[pp,[de,su,venta]]],
  [[1,3],[ppn,[a,'Iberia']]],
  [[1,4],[pc,[,]]],
  [[1,5],[pn,['VIASA']]],
  [[1,6],[vc(fin),[contaba,con]]],
  [[1,7],[np,[ocho,aviones]]],
  [[1,8],[pc,[',']]],
  [relative,
    [[2,1],[cm(que),[que]]],
    [[2,2],[vc(fin),[tenían]]],
    [[2,3],[pp,[en,promedio,13,años,de,vuelo]]],
    [[2,4],[pc,[',']]]]]].
```

7. Clause Analyzer

The function of the clause analyzer is to take a clause length sequences of phrases and identify the grammatical dependency relations among the phrases contained, outputting a set of tables each of which represents a possible dependency analysis of the clause.

Given an input sentence marked for clause boundaries, the component iteratively applies an as yet limited DCG to each clause. Basically, the DCG states that a clause consists of a subject followed by a finite verbal construction, optionally followed by an object, optionally followed by some number of adverbial modifiers or a clause consists of a passive participial optionally followed by an object.

Subjects and objects consists of a noun phrase, optionally followed by a prepositional phrase, or a preposition plus proper noun. Modifiers consist of a prepositional phrase or a preposition plus proper noun.

Applying this DCG to the input above results in an output of the following simplified form:

```
[[finite,
  [[1,1],[pp,['Al',momento]]],

  [[1,8],[pc,[',']]],
  [[[1,6],[1,6],head],
  [[1,5],[1,6],subj],
  [[1,7,],[1,6],obj]],
  [relative,
  [[2,1],[cm(que),[que]]],
    ...
    [[2,4],[pc,[',']]],
    [[[2,2],[2,2],head],
     [[2,3],[2,2],pmod]]]]]
```

The dependency relations have been captured in the tables appended to the end of each clause.

There is a second pass over each clause during which specific subgoals of the rules are ignored. So, for example, the identification of *en promedio de 13 años de vuelo* as a circumstantial modifier of *tenían* was made on the basis of applying one of clause rules in the DCG with the subject subgoal suspended.

8. Reading Ranker

The Reading Ranker provides a ranked listing of the possible input sentence readings produced by the Phrase Analyzer and Clause Analyzer. Essentially, the Reading Ranker must search and rank the space of all possible syntactic and semantic combinations.

In light of this combinatorially large search space, an attempt must be made to constrain its size. Currently, the use of both syntactic and semantic constraints are being considered. Syntactically, only the best parses from the Clause Analyzer can be evaluated. Also, part-of-speech tags can be used to reduce the number of possible word senses. Semantically, preference information should also eliminate or indicate the likelihood of certain word sense co-occurrences.

It is unlikely that the above constraints will sufficiently reduce the size of the search space so as to allow the use of conventional (exhaustive) search methods. Thus, weak search techniques, which do not exhaustively search the entire space, must be employed.

(Cowie, Guthrie and Guthrie, 1992) investigated simulated annealing for the purpose of word sense disambiguation. Their approach involved disambiguating all of the words in a sentence simultaneously, where the rank (evaluation) of a particular set of selected senses was determined by the Overlap of their LDOCE definitions. They report a word disambiguation accuracy of 47% at the sense level and 72% at the homograph level.

The LDOCE overlap disambiguation method described above was re-implemented with a genetic algorithm replacing simulated annealing as the weak search technique, with almost no change in accuracy. Part-of-speech information was then used to restrict the possible senses for the words. Not surprisingly, homograph level accuracy jumped to above 90%, due to the fact that in LDOCE, the senses of many words are grouped into homographs on the basis of their part of speech. Sense level accuracy, on the other hand, actually dropped by several points. In part this decrease in performance was due to an inadequate morphological analyzer which produced stems with incorrect parts of speech.

It is likely that the less than satisfactory results of both simulated annealing and the genetic algorithm are the result more of an ineffective evaluation metric than of an inherent inability to search this particular space. In fact, several experiments revealed that the best interpretation was being evaluated as much worse by the LDOCE evaluation metric than incorrect interpretations. The possibility of enhancing the LDOCE overlap method by including WordNet synsets of the sense definition words is being considered.

A certain degree of skepticism remains as to whether dictionary sense definition overlap or co-occurrence can be used to successfully disambiguate word senses. Thus, an investigation has been launched to determine whether statistically derived word sense meaning vectors might prove to be more successful. This approach has been used by (Schütze, 1993) and (Landauer, 1994) to disambiguate words and discriminate amongst synonyms. These vectors might be used in one of two ways: either as a more accurate evaluation metric for a weak search technique, or in a more direct word for word disambiguation attempt.

Summary and System Performance

The Panglyzer functions as the analysis portion of a knowledge-based MT engine. The results of this KBMT system are themselves only inputs to a multi-engine MT system (Pangloss). As a result, it is difficult to judge the performance of the Panglyzer solely on system (Pangloss) throughput, and we have yet to develop a notion of adequacy of analysis (Panglyzer) output. However, each module of the Panglyzer has a fairly well-defined task, and appropriate output for each module can be judged with a fair degree of accuracy.

For instance, the Preprocessor is able to identify sentence- and word- boundaries with near perfect accuracy. The Part-of-Speech Tagger operates at about 93% accuracy, when compared with the judgments of Spanish language experts. The Phrase Recognizer has an accuracy rate of 90% on Part-of-Speech Tagger output. Discounting Tagger errors its accuracy rate is roughly 98%. The Proper-Noun Classifier will classify about 80% of the proper nouns in a given text correctly.

The remainder of the module have yet to undergo rigorous testing. For the Phrase Analyzer, a performance estimate on the basis of a sample text produced an appropriate representation for 77% of the phrases in the text. This representation may be one of several produced for any particular phrase. When failure due to missing lexical items and incorrectly recognized phrases is discounted, appropriate representations were produced for 97% of the phrases. As is evident, the quality of the output of the Phrase Analyzer is highly dependent on having lexicon entries for the words in the text being analyzed.

Over a short text the Clause Recognizer identified about 76% of the clauses and 56% of the clauses contained the appropriate constituents. In a test over a text with 40 clauses, the Clause Analyzer produced correct and complete results in only 4 cases, partially correct in 25, and incorrect results in 11.

Finally, given the cascading architecture described here, the performance of any module usually depends greatly on the performance of the preceding modules. For example, an incorrect tag may well cause an incorrect phrase to be constructed, which may then be analyzed and grouped into a clause incorrectly.

To account for these difficulties, we are attempting to measure the performance of each module in two respects: first using actual system output from previous modules and second on the basis of manually corrected "golden" input.

References

Cowie, J., Guthrie, J. and Guthrie, L., 1992, Lexical Disambiguation using Simulated Annealing, *Proceedings of the 15th International Conference on Computational Linguistics (COLING-92),* Nantes, France, July, pp. 359-365.

Huang, X-M, 1988, XTRA: The Design and Implementation of a Fully Automatic Machine Translation System. *Memoranda in Computing and Cognitive Science,* MCCS-88-121, Computing Research Laboratory, New Mexico state University, Las Cruces, New Mexico.

Landauer, T. K., 1994, "How is it that you know so much?", An Invited lecture at New Mexico State University, January 1994.

Pereira, F. and Warren, D., 1980, Definite Clause Grammars for Language Analysis--A Survey of the Formalism and A Comparison with augmented Transition Networks. *Artificial Intelligence,* 13: 231-278.

Procter, P., et al, 1978, *Longman Dictionary of Contemporary English.* Longman Group Limited, Harlow, Essex, England.

Schütze, H., 1993, Word Space. In S. J. Hanson, J. D. Cowan, and C. L. Giles (Eds.), *Advances in Neural Information Processing Systems 5,* 895-902. San Mateo CA: Morgan Kaufmann Publishers.