Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger

Dan Tufis
The Romanian Academy Centre for Artificial Intelligence
Bucharest, Romania
[tufis@racai.ro]

Oliver Mason
The University of Birmingham, United Kingdom
[O.Mason@bham.ac.uk]

Abstract

This paper describes an experiment on tagging Romanian using QTAG, a parts-of-speech tagger that has been developed originally for English, but with a clear separation between the (probabilistic) processing engine and the (language specific)resource data. This way, the tagger is usable across various languages as shown by successful experiments on three quite different languages: English, Swedish and Romanian. After a brief presentation of the QTAG tagger, the paper dwells on language resources for Romanian and the evaluation of the results. A complexity metrics for tagging experiments is proposed which considers the performance of a tagger with respect to the "difficulty" of a text.

Introduction

Lexical ambiguity resolution is a key task in natural language processing (Baayen & Sproat, 1996). It can be regarded as a classification problem: an ambiguous lexical item is one that in different contexts can be classified differently and given a specified context the disambiguator/classifier decides on the appropriate class. The features that are relevant to the classification task are encoded into the tags. It is part of the corpus linguistics folklore that in order to get high accuracy level in statistical POS disambiguation, one needs small tagsets and reasonable large training data. The effect of tagset size on tagger performance has been discussed in (Elworthy, 1995); what a reasonable training corpus means, typically varies from 100,000 words to more than one million. Although some taggers are advertised to be able to learn a language model from raw (unannotated) texts, they require a post validation of the output and a bootstrapping procedure that would eventually get the tagger (in possibly several iterations) to an acceptable error rate. The larger the tagset, the larger the necessary training corpora (Berger, Pietra & Pietra, 1996). Provided that enough training data is available, this should not be too problematic as long response time is seriously affected. It is quite obvious that language models based on large tagsets would need large memory resources and given current hardware limitations, a lot of overhead (required for memory management) would decrease the performance to an unacceptable level.

Tiered Tagging

Most of the real time taggers (if not all of them) are able to ensure a fast response due to their ability to keep the language model in the core memory. In case there is not enough RAM to load it, a typical tagger (at least those in the public domain) would give up (either graciously informing about lack of memory, or just frustratingly crashing). For instance, with about 700 tags, a language model of 350 MB or more would not be surprising (this was the actual size of the transition matrix while training QTAG for Romanian) and it would be out of question to keep it in RAM on usual computers,. Apparently there are two solutions for overcoming such a deadlock: either to reduce the tagset to a manageable size and lose information or to modify the tagger with some extra-code, to take care of data swapping and accept a probably serious degradation of response time. In (Tufis, 1998) it is argued that there is another solution providing a nice compromise. With a small price in tagging accuracy (as compared to a reduced tagset approach), and practically no price in computational resources, it is possible to tag a text with a large tagset by using language models built for reduced tagsets and consequently small training corpora. We call this way of tagging *tiered tagging.* Given the space limitation we will not go into details concerning tiered tagging (a full presentation of the tiered tagging approach and an in-depth discussion of the methodology and results are given in (Tufis, 1998)). In general terms, tiered tagging uses a hidden tagset (we call it C-tagset) of a smaller size (in our case 89 tags) based on which a language model is built. This language model serves for a first level of tagging. Then, a post-processor deterministically replaces the tags from the small tagset with one or more (in our experiments never more than 2) tags from the large tagset (we call it MSD-tagset). The words that after this replacement become ambiguous (in terms of the MSD-tagset annotation) are more often than not the difficult cases in statistical disambiguation. Therefore, if full disambiguation is desired (instead of k-best tagging), the different interpretations of the few words that remain ambiguous (in our experiment, less than 10%) are differentiated by very simple contextual rules. These rules investigate, depending on the ambiguity class, left, right or both contexts within a limited distance (in our experiment never exceeding 4 words in one direction) for a disambiguating tag or wordform. The success rate of this second phase was higher than 98%. Given the rare cases when contextual rules application is required, the response time penalty is very small. Depending on how accurate the contextual rules are, the error rate for the final tagged text could be practically the same as for the hidden tagging phase. Obviously, the reduced and the extended tagsets have to be in a specific relation (the small tagset should subsume the large one). In (Tufis, 1998) it is shown how a reduced tagset

(C-tagset) can be interactively designed from a large tagset (MSD-tagset), based on a trial-and-error ID3-like procedure. The global error rate of the tiered tagging is given by the relation:

$$\text{Error-rate} = (N_{errors\_tagger} + N_{errors\_mapping})/N_{words}$$

where:

N_{words} is the total number of words in the tagged text

N_{errors_tagger} is the number of errors made during the first phase of tagging (C-tagset tagging)

$N_{errors\_mapping}$ is the number of errors made by the second phase of tagging (the mapping from C-tagset to MSD-tagset)

As any error made at the first step would show up in the final result, the **Error-rate** above is cumulatively specified. The additional errors, **$N_{errors\_mapping}$**, could be roughly estimated as follows (for a rigorous upper-limit evaluation of this estimation, see (Tufis, 1998)): Let us consider that all the remaining ambiguous words **N_{amb}**, were correctly tagged in the first step (this is certainly an overestimation if one considers a normal distribution of errors). Then, considering e as the average error rate for rule-based disambiguation, it follows that approximately $N_{amb}*\varepsilon$ words will be improperly disambiguated in the second phase. The C-tagset induction process grants a (user specified) maximum value for the N_{amb}/N_{words}. In our experiment this was set to 10%, that is a C-tagset tagged text can be mapped error-free onto a MSD-tagset tagged text with at most 10% words remaining ambiguous. With an e less than 5%, one gets less than 0.5% error contribution of the C-tagset onto MSD-tagset mapping in the final accuracy of the MSD-tagset tagged text.

Given that the final accuracy of tiered tagging depends on the performance of the first (hidden) tagging step, we will address here only this phase. For the second phase of the tiered tagging (including C-tagset design, contextual rules acquisition and application and commented results), the interested reader is pointed to (Tufis, 1998).

The tagger

There are two basic approaches to part-of-speech tagging: rule-based and probabilistic. It is also possible to combine both into a hybrid approach. The tagger presented in this paper is purely probabilistic.

The first step in any tagging process is to look up each token of the text in a dictionary. If the token cannot be found, the tagger has to have some fallback mechanism, such as a morphological component or some heuristic methods. The difficult task is to deal with ambiguities: only in trivial cases will there be exactly one tag per word. This is were the two approaches differ: while the rule-based approach tries to apply some linguistic knowledge (usually encoded in rules) in order to rule out illegal tag combinations, a probabilistic tagger determines which of the possible sequences is more probable, using a language model that is based on the frequencies of transitions between different tags.

A rule-based language model can be created by a human using linguistic knowledge, but it is not reasonable to hand-coded a probabilistic language model. These models are generally created from training data, i.e. they are learnt from examples. This is the way QTAG works: it uses only probabilities for disambiguating tags within a text, and no rule-based mechanism. As a result, it can easily adapted for new languages, as long as some pre-tagged training corpus is available. The tagger has been developed at Corpus Research in Birmingham and is freely available for research purposes (for more information on this see http://www-clg.bham.ac.uk/tagger.html).

The basic algorithm is fairly straight-forward: at first, the tagger looks up the dictionary for all possible tags that the current token can have, together with their respective lexical probabilities (i.e. the probability distribution of the possible tags for the word form). This is then combined with the contextual probability for each tag to occur in a sequence preceded by the two previous tags. The tag with the highest combined score is selected. Two further processing steps also take into account the scores of the tag as the second and first element of the triplet as the following two tokens are evaluated.

QTAG works by combining two sources of information: a dictionary of words with their possible tags and the corresponding frequencies and a matrix of tag sequences, also with associated frequencies. These resources can easily be generated from a pre-tagged corpus.

The tagging works on a window of three tokens, which is filled up with two dummy words at the beginning and the end of the text. Tokens are read and added to the window which is shifted by one position to the left each time. The token that 'falls' out of the window is assigned a final tag. The tagging procedure is as follows:

1. read the next token
2. look it up in the dictionary
3. if not found, guess possible tags
4. for each possible tag
   a. calculate $P_w = P(\text{tag}|\text{token})$ the probability of the token to have the specified tag
   b. calculate $P_c = P(\text{tag}|t_1,t_2)$, the probability of the tag to follow the tags t_1 and t_2.
   c. calculate $P_{w,c} = P_w * P_c$, the joint probability of the individual tag assignment together with the contextual probability.
5. repeat the computation for the other two tags in the window, but using different values for the contextual probability: the probabilities of the tag being surrounded and followed by the two other tags respectively.

For each recalculation (three for each token) the resulting probabilities are combined to give the overall probability of the tag being assigned to the token. As these values become very small very quickly, they are represented as logarithms to the base 10. For output, the tags are sorted according to their probability, and the difference in probabilities between the tags gives some measure of the confidence with which the tag ought to be correct (see below for an example of this).

The tagger is implemented in a client-server model. The server is implemented in C, while the client is written in Java. The knowledge base resides on the server and the tagging engine retrieves the relevant data for tags and tag sequence probabilities from it via a network connection. This way, the tagger can run on different platforms, provided there is a server available. The total size of the executable code files is less than ten kilobytes. Furthermore, since the tagger is implemented as a Java class, it can easily be used as a module in other linguistic applications.

The non-ASCII characters dealt with by QTAG are encoded as standard SGML character entities as the server is not Unicode capable (but the client is).

The guesser

The morphology necessary to deal with the unknown words is encoded as a language specific resource. There are two different guessers: the first one is based on a list of the final three letters of all words from the lexicon with their respective tag probabilities. This list is built automatically during the training of the tagger. This very simple mechanism seems sufficiently general to yield good results at little cost, even in languages with a rather complex morphology.

The second guesser (Romanian-specific), built at RACAI, is more linguistically motivated and considers the real inflectional endings for open class words (nouns, adjectives, verbs). Each ending (including the 0-ending) is associated with an ambiguity class consisting of appropriate tags for open class words (the 0-ending includes also tags for abbreviations, residuals, and interjections). By a retrograde analysis (right to left) of the unknown word, the guesser identifies all possible endings. The ambiguity classes corresponding to all the possible endings are merged, with higher probability assigned to the interpretations provided by longer endings. Depending on the way the guesser is invoked, the unknown word is assigned either this merged ambiguity class(the default) or the ambiguity class corresponding to the longest matched ending. To evaluate the guesser, we extracted from our main dictionary (D0) all the words which contained in their ambiguity class an interpretation belonging to close classes, words with root 2 characters long and also some irregular open class words and created a guesser lexicon of about 4000 entries (D1).The rationale for including the words with roots 2 characters long in D1 was because we imposed the guesser the restriction that the remaining part of a word, after removing the longest ending, be longer than 2 letters. The first experiment considered testing the guesser in the "all interpretations"-mode on the words in D0-D1. All the wrongly classified words were analysed and a few idiosyncratic endings and interpretations were added to the list of <*ending : ambiguity-class*> pairs. Some irregular words were also moved into D1. This step was repeated until a precision of 100% was obtained. A step further was to evaluate the effectiveness of the "longest match" heuristics, that is we set the guesser to return the ambiguity class for the longest identified ending only. The rationale for this was that in the vast majority of cases, recognising longer endings made the previous ones quite unlikely. The rare exceptions were put in the D1 lexicon. The results we obtained by running the guesser in the "longest match" mode on D1-D0 (more than 400,000 wordforms) exceeded our expectations. There were reported only 8892 "errors" (2.17%). By error analysis we found that almost half of these errors were not guesser errors: they were either entries in the dictionary having assigned a wrong MSD or entries corresponding to fake words that appeared in Orwell's *1984* (words belonging to *newspeak* or *proles* speak). The real errors (4324, or 1.08%) clustered quite regularly and pointed out a small number of words with the final letter(s) in their roots combining with the real endings. Put differently, the errors were given by wrong segmentations root+ending: $\alpha+\beta\gamma$ instead of $\alpha\beta+\gamma$ (both γ and $\beta\gamma$ being recorded as endings). Given the few instances of such combinations, one way to overcome this type of error was to import those MSDs associated to γ which are valid for the root $\alpha\beta$ into the interpretation list associated with the ending $\beta\gamma$. The average length of an MSD-ambiguity class returned for an unknown word was 11. Given that the guesser is supposed to work for the tagger, we turned the MSDs into tags as used by the tagger (see section "The Tagset") and the average length of tag-ambiguity class decreased to 3 tags per unknown word. We run again the guesser over the D0-D1 lexicon and we got(in the "longest match" mode) a much smaller number of errors: 1302. In principle, with less than 0.3% guessing errors we could have been satisfied. However we made another step that was very easy to implement, could ensure almost error-free guessing and would not alter the computational and accuracy performance of the tagger. What we noticed was that for 968 words that were not correctly guessed using the "longest match" mode, the correct interpretation could be found in the second longest matched endings. So, the guesser was set to return the union of the interpretations for the 2 longest matched endings. With this modification, the average length of the tag-ambiguity class doubled (6 tags/unknown words). Given that the ambiguity classes very frequently included tags that subsumed some other tags in the same list, and also that quite frequently there were cases when two or more tags in the ambiguity class would represent the full expansion of a more general tag, we decided to eliminate all these redundant tags from the their ambiguity classes. Consequently, the average lengths of the ambiguity class returned for an unknown word become reasonably shorter (3.3 tags). The number of errors returned by the guesser run in this mode for the words in D0-D1 was 181 (0.04%) but all the wrongly labelled words were infrequent wordforms, unlikely to occur other than as hapaxes in normal texts.

Language Resources

The Romanian wordforms lexicon was created based on a 35.000-lemma lexicon by means of our EGLU natural language processing platform (Tufis, 1997). Since several words in the corpus were not in the EGLU lexicon, most of them were manually lemmatised, introduced in the unification-based lexicon and later on expanded to the full paradigms of every new lemma. The table in Figure 1 provides information on the data content of the main dictionary that is used for the corpus analysis.

The MSDs (Morpho-Syntactic Descriptions) represent a set of codes as developed in the MULTEXT-EAST project[1]. AMB-MSD represents the number of ambiguity classes (Weischedel & all, 1993; Abney, 1997) or genotypes (Tzoukermann & Radev). The morpho-syntactic descriptions are provided as strings, using a linear encoding. In this notation, the position in a string of characters corresponds to an attribute, and specific characters in each position indicate the value for the corresponding attribute. For a given wordform several MSDs might be applicable (accounting this way for homographs). The set of all the MSDs applicable to a given word defines the MSD-ambiguity class for that word. The Romanian lexicon contains 869 MSD-

[1] For the final reports see http://nl.ijs.si/ME

ambiguity classes (for more details on the Romanian dictionary and corpora encoding and several relevant statistics see (Tufis & all, 1997)).

Entries	Word-forms	Lemmas	MSDs	AMB-MSD
419869	347126	33515	611	869

Figure 1: Romanian dictionary overview

The corpus used in the experiments and evaluation reported here was made of the integral texts in two books: Orwell's *1984* and Plato's *The Republic*. A brief overview of these texts is given below:

Text	Occurrences	Words	Lemmas	MSDs	AMB-MSD
1984	101460	14040	7008	396	524
Republic	114720	10350	4697	369	490
Common	56804	4016	2524	319	394

Figure 2: Romanian corpus overview

The existent SGML markup (TEI conformant) was stripped off and the text has been tokenised. Please note that a token is not necessarily a word: one orthographic word may be split into several tokens (the Romanian "da-mi-l" (give it to me) is split into 3 tokens) or several orthographic words may be combined into one token (the Romanian words "de la" (from) are combined into one token "de_la"). Each lexical unit in the tokenised text was automatically annotated with all its applicable MSDs and then hand disambiguated. This was the main resource, which we called *MSD-tagged corpus*, for training and evaluating the tagger. The figure below exemplifies the MSD-tagged corpus.

Într-	Spsay
o	Tifsr
zi	Ncfsrn
senină	Afpfsrn
şi	Ccssp
friguroasă	Afpfsrn
de	Spsa
aprilie	Ncms-n
,	COMMA
pe	Spsa
când	Rw
ceasurile	Ncfpry
băteau	Vmii3p
…	

Figure 3: MSD-tagged corpus overview

The Tagset

The tagset for Romanian contains 79 tags for different morpho-syntactic categories, plus 10 tags for punctuation. The tagset has been derived by a trial-error procedure from the 611 morpho-syntactic description codes(MSDs) defined for encoding the Romanian lexicon. By observing the MSD clustering in the ambiguity classes, we started to eliminate several attributes in the MSD codes that would reduce the cardinality for the MSD ambiguity classes. If one thinks an MSD in terms of a (flat) feature structure, then this elimination of attributes in a given MSD represents a generalisation of the corresponding MSD that would subsume the initial one.

The generalisation process observed the following general guidelines:

- eliminate the attributes that are unambiguously recovered by looking up the word in the lexicon;
- eliminate the attributes with little influence on the distribution of the words in the same POS class;
- eliminate the attributes which do not influence the graphic form of the words in the given POS class;
- preserve the attributes which correlates in co-occurring words.

The first step in the tagset design was to keep in the tagset only the POS information, train the tagger on this minimal tagset and observe the errors made by the tagger on a few paragraphs randomly extracted from our corpus. The errors, signaled in the form of Cat_a *instead of* Cat_b allowed us to build up the confusion sets for the minimal tagset:(Cat_1 Cat_2...Cat_m) *instead of* Cat_b. For all the wrongly marked up words we extracted the context and identified the attributes that were likely to help the tagger to make the correct choice. We added these attributes to the tags, repeated the training and tagging and produced each time another list of confusion sets. As a result of this iterated step, each MSD-ambiguity class ($MSD_1MSD_2...MSD_k$) was equivalated to a C-TAG ambiguity class (tag_1 tag_2 ... tag_i) with $i \leq k$ and a many-to-one mapping between MSDs and C-TAGs was defined: MSD_{n1} $MSD_{n2}...MSD_{np} \rightarrow tag_m$ (or vice versa a one-to-many mapping $tag_m \rightarrow MSD_{n1}MSD_{n2}...MSD_{np}$). When this repeated process stabilised, we checked for the MSD recoverability degree of a given C-tag annotated text. This test is very simple and it computes for a token tok_i disambiguated as tag_k the intersection between the ambiguity class of tok_i and the mapping class of tag_k. If this intersection contains only one element, then the token tok_i is fully MSD-recoverable when it is tagged with tag_k. Otherwise, the token tok_i is considered partially recoverable. Defining recoverability degree of a given C-tag annotated text as the number of fully MSD-recoverable tokens per total number of tokens, we reached after several adjustments of the tagset *and the dictionary information* a 90% recoverability degree of the test corpus. The adjustment of dictionary information consisted in conflating some interpretations of a few functional words which were hard to distinguish and removing a few distinctions that were impossible to be made on a purely statistical and syntactical basis. The most obvious case was the distinction between numeral and article readings for the words *un* (*a/an* or *one* -masculine) and *o* (*a/an* or *one* -feminine). To identify the numeral readings of these words some semantic criteria would have been necessary (for instance to define a subclass of nouns "measure unit"), so we decided to classify them only as articles.

Let us consider a few examples to clarify this methodological procedure. For the categories Noun Adjective, the attributes Case and Number were preserved since when the latter modifies the former they must agree in Case and Number. Since in many cases the nouns and adjectives are ambiguous with respect to these attributes when considered in isolation, but rarely when considered in collocation, preserving these attributes proved to be extremely

helpful for the performance of the tagger. On the other hand, Gender (which also is subject to the agreement requirement) is very rarely ambiguous. The Gender of a noun, adjective, pronoun, determiner, article, numeral, or participle is almost always recoverable from the wordform itself, and in those very rare cases when it is not, the immediate context (plus the agreement rule) does the job. We have so far not encountered (in our corpus) a single instance where this was not the case. Therefore, preserving the Gender was not relevant and after its removal the number of tags decreased significantly (and the accuracy of the tagger increased accordingly). The definiteness attribute is also fully recoverable from the wordform, but given the grammatical constraints, keeping it was very useful in helping the tagger to discriminate among nouns, adjectives and participles (ex: *frumosul baiat* versus *frumosul baiatului* or *iubitul baiat* versus *iubitul baiatului* or *baiatul iubit*). With the verbs, the distinction was preserved between finite (main and auxiliary) verbs nonfinite verbs (infinitive, participle and gerund). The finer distinction between main and copulative uses (valid for a few verbs) was a constant source of tagging mistakes so we removed it from our lexical encoding, and consequently it disappeared from the tagset. For the finite verbs the attribute Person was preserved and just only for the auxiliaries the Number attribute was kept. Eliminating the tense attribute dramatically decreased the number and the cardinality of the ambiguity classes. Eliminating the Mood attribute further decreased the number and the cardinality of the ambiguity classes.

The most troublesome categories were the pronoun and the determiner (this class contains what grammar books traditionally call adjectival pronouns and due to several commonalties it was merged with the pronominal class). As with the nominal categories, we opted for preserving the Case and Number attributes. However, with only these two attributes the tagset design methodology classified most of the pronominal MSDs into functionally and distributionally unrelated classes. The only exceptions were the reflexive pronouns and determiners in accusative or dative, which clustered together (they are explicitly marked for these cases and are insensitive to number). The next step was to consider the types of pronominals and determiners in the classification. With this new feature, the number of tags remained the same, but their recoverability degree slightly improved and the clustering of MSDs was more linguistically motivated.

Similar considerations applied for all the other word classes and the resulted tagset (initially 73 proper tags and 10 punctuation tags) was used in a more comprehensive evaluation of the tagging process. The evaluation of the tagger performance (described in another section below) was done on a larger text by two independent experts. Based on the insights gained from the analysis of the errors, the tagset was slightly modified (79 proper tags and 10 punctuation tags) and some dictionary entries were changed. The final tagset and dictionary modifications are discussed in the section on Evaluation.

The Training Corpus

The training corpus (CTAG-corpus) was obtained from the MSD-tagged corpus by substituting the MSDs with their corresponding corpus tags (using *the MSD to C-tag mappings* described in the previous section). The figure below shows a sample of the CTAG corpus (based on the final tagset, discussed in the next section). Although the two texts[2] contain altogether more than 250.000 words, the number of different words is just 20,384 (less than 6% of the number of wordforms in the lexicon). However, for the distributional analysis and morpho-syntactical disambiguation purposes the selected texts offer enough evidence to extract reliable data and draw realistic conclusions. Out of the 611 MSDs defined in the lexicon, these texts contain 444 MSDs (72.66%). From the 869 MSD-ambiguity classes, these two texts contain 620 (71.13%). This is to say that most of the words raising ambiguity problems appeared at least in one of the two texts.

Într-	S
o	TSR
zi	NSRN
senină	ASRN
şi	CVR
friguroasă	ASRN
de	S
aprilie	NSN
,	COMMA
pe	S
când	R
ceasurile	NPRY
băteau	V3
…	

Figure 4: C-TAG corpus overview

The Training Process

Out of the training corpus, 90% was retained for the proper training and the rest of 10% (the first parts of both *1984* and *The Republic*) was used for the validation purposes. The training process is merely are formatting exercise. The data contained in the training corpus is sorted and filtered twice, once to extract the lexicon (if it does not exist already) and once for the tag trigrams. The extraction of the relevant information is done mostly with standard Unix text processing tools and two special purpose programs. As the final step, the three letter word-endings are extracted from the lexicon and a 'guess list' is created from them. This guess list is used as a guesser resource unless another guesser is installed (see above).

The tagger is ready for use with the new resources as soon as the data files are accessible to the server program. The output format of the tagger is a vertical text with all possible tags; the tags are assigned a probability value and they are sorted, so that the most likely tag comes first (see example below[3]).

```
…
o   [TSR:-4.799][PPSA:-7.101][QF:-9.839][VA3S:-11.481][I:-11.886]
zi  [NSRN:-1.066][V2:-10.669]
…
```

[2] To cope with unknown words, additional sentences were introduced in the corpus so that all the ambiguity classes corresponding to the endings used by the guesser be taken into account when constructing the language model.

[3] The scores are expressed as logarithms, so the smaller the (signed) value of the assigned score, the smaller the probability.

This allows expressing a judgment on the confidence with which the tag has been assigned: if the probability difference between the first tag and the next tag(s) is comparatively large (as it is for *zi* in the example above), the decision is more certain than in case of near equal probabilities. Apart from assessing the confidence, this can also be used as a starting point for manual correction of the text, as those words with similar probabilities are more likely to contain errors. The evaluation with Romanian data has shown that with most errors the difference between the wrongly assigned tag and the correct tag was rather small, whereas it was much bigger with correctly assigned tags.

Evaluation and the final tagset

The tagger was trained and evaluated several times on different segments of the hand-disambiguated corpus with various results. This was against our expectations so we made a pretest, training the tagger on the whole corpus and running it on the same data. As one would expect, the accuracy was very high, but the error analysis confirmed one of our suspicions: out of the reported errors there were almost 500 (non-systematic) errors (189 in *"1984"* and 301 in *"The Republic"*) made by the human disambiguators in the process of building up the MSD-tagged corpus. The tags attributed by the tagger were in all these cases the correct ones. They were corrected, the tagger retrained and the test redone. With data corrected, there were reported a few more human errors (8 in "*1984*" and 11 in "*The Republic*"). A third trial of this test didn't revealed new human-made errors (but there is of course no certainty that they were all discovered) so, the tagger was trained on three texts, building three language models. The first training was done on 90% of *"1984"*, the second on 90% of "*The Republic*" and the third on the concatenation of the texts used in the first two (90% of each of the two books). The resulting language models were used to test the corresponding unseen 10% of the texts. The error analysis suggested some modifications of the tagset and of a few entries in the dictionary.

data /LM	no. of words	no. of errors	accuracy
10% *1984* / 90% 1984 LM	11791	257	97.82%
10% *Republic* /90% Rep LM	13696	533	96.10%
10% 1984 and Republic / 90%(1984+Rep) LM	25487	1112	95.63%

Figure 5: Initial tagging results

The basic test on the unseen part of the training corpus provided the results shown in Figure 5.

As shown in the previous sections, the tagger output is a tabular format with each word on a line followed by an ordered list of pairs *[tag:probability]*. The above mentioned accuracy was measured considering only the tag with the highest probability. In case the tagger was definitely wrong (that is the probability of the chosen tag was much larger than the probability of the correct one) we manually modified the MSD and the corresponding tag. This happened for instance with the word-forms *si* and *si-* which in the lexicon are listed as reflexive pronoun (*himself/herself*), conjunction (*and*) and adverb (*still, yet*). While the reflexive pronoun reading was almost always correctly assigned, the other two interpretations (C and R) were constantly confused. This distinction is in many cases very difficult to be made even by native speakers, so we merged them into one tag CVR (to be read as conjunction with adverbial instances). Another example is given by the word *fi*, initially encoded in the lexicon as an infinitive (*to be*) and as an aspectual particle. Since the particle reading rarely has been correctly identified by the tagger and since this is a very frequent word in Romanian, its "contribution" to the errors list was significant (4.28% of all the errors). We removed from the lexicon the particle interpretation leaving only the one for infinitive (what most grammar books would do), thereby removing this source for errors. Another follow-up modification of the tagset was to make a distinction among the particles (QN, QS, QF) initially conflated into one tag (Q). This distinction eliminated several errors (such as distinction between auxiliary and main usage of some verbs). Furthermore, making a distinction (which the initial tagset did not have) between proper nouns and common nouns helped in reducing tagging errors of the word "*lui*" which is always a genitival article when precedes a proper noun, as opposed to being a pronoun otherwise (which is by far the most probable tag). Another problem was made by the initial tag ASRN which was meant for adjectives, singular, direct case, indefinite. By analysing all the MSDs that were mapped to this tag we noticed that they identified only feminine adjectives, although we didn't mean to preserve the gender distinction in our tagset. This tag was in most cases mistakenly preferred to the more general tag ASN (adjectives, singular, indefinite). Again, by observing the MSDs mapped onto the ASN tag, we noticed only singular, masculine, indefinite adjectives which are Case undetermined. Therefore, a natural decision was to conflate the ASRN and ASN tags. This decision eliminated tagging errors for those few but frequent adjectives which have identical forms for both masculine and feminine gender when singular or indefinite (for instance "*mare*"-*big*). The actual tagset is briefly described below.

A Adjective
AN Adjective, indefinite
APN Adjective, plural, indefinite
APON Adjective, plural, oblique, indefinite
APOY Adjective, plural, oblique, definite
APRY Adjective, plural, direct, definite
ASN Adjective, singular, indefinite
ASON Adjective, singular, oblique, indefinite
ASOY Adjective, singular, oblique, definite
ASRY Adjective, singular, direct, definite
ASVN Adjective, singular, vocative, indefinite
ASVY Adjective, singular, vocative, definite
C Conjunction
CVR Conjunction or Adverb
I Interjection
M Numeral
NP Proper Noun
NN Common Noun, singular
NPN Common Noun, plural, indefinite
NPOY Common Noun, plural, oblique, definite
NPRN Common Noun, plural, direct, indefinite
NPRY Common Noun, plural, direct, definite
NPVY Common Noun, plural, vocative, definite
NSN Common Noun, singular, indefinite

NSON Common Noun, singular, oblique, indefinite
NSOY Common Noun, singular, oblique, definite
NSRN Common Noun, singular, direct, indefinite
NSRY Common Noun, singular, direct, definite
NSVN Common Noun, singular, vocative, indefinite
NSVY Common Noun, singular, vocative, definite
NSY Common Noun, singular, definite
PI Quantifier Pronoun or Determiner
PXA Reflexive Pronoun, accusative
PXD Reflexive Pronoun, dative
PSP Pronoun or Determiner, poss or emph. plural
PSS Pronoun or Determiner, poss or emph, singular
PPPA Personal Pronoun, plural, acc., week form
PPPD Personal Pronoun, plural, dative
PPSA Personal Pronoun, singular, accusative
PPSD Personal Pronoun, singular, dative
PPSN Personal Pronoun, singular, nom., non-3 person
PPSO Personal Pronoun, singular, oblique
PPSR Personal Pronoun, singular, direct
PPPO Personal Pronoun, plural, oblique
PPPR Personal Pronoun, plural, direct
RELO Pronoun or Determiner, relative, oblique
RELR Pronoun or Determiner, relative, direct
DMPO Pronoun or Determiner, dem., plural, oblique
DMSO Pronoun or Determiner, dem., singular, oblique
DMPR Pronoun or Determiner, dem., plural, direct
DMSR Pronoun or Determiner, dem., singular, direct
QN Infinitival Particle
QS Subjunctive Particle
QF Future Particle
QZ Negative Particle
R Adverb
S Preposition
TP Article, indefinite or possessive, plural
TPO Article, non-possessive, plural, oblique
TPR Article, non-possessive, plural, direct
TS Article, definite or possessive, singular
TSO Article, non-possessive, singular, oblique
TSR Article, non-possessive, singular, direct
V1 Verb, main, 1st person
V2 Verb, main, 2nd person
V3 Verb, main, 3rd person
VA1 Verb, auxiliary, 1st person
VA1P Verb, auxiliary, 1st person, plural
VA1S Verb, auxiliary, 1st person, singular
VA2P Verb, auxiliary, 2nd person, plural
VA2S Verb, auxiliary, 2nd person, singular
VA3 Verb, auxiliary, 3rd person
VA3P Verb, auxiliary, 3rd person, plural
VA3S Verb, auxiliary, 3rd person, singular
VG Verb, gerund
VN Verb, infinitive
VP Verb, participle
X Residual
Y Abbreviation

data /LM	no. of words	no. of errors	accuracy
10% *1984* /90% 1984 LM	11791	189	98.39%
10% *Republic*/90% Rep LM	13696	393	97.13%
10% 1984 and Republic/ 90%(1984+Rep) LM	25487	963	96.22%

Figure 6: Final tagging results

The tagger evaluation was repeated for this final tagset and the results (which improved significantly) are shown in the table in Figure 6.

A Complexity Metric for Tagging Experiments

The performance of a tagger is usually measured in the percentage of correct tag assignments. While this initially sounds quite plausible, it does not say very much about the quality of the tagger. There are parameters that influence the performance which are not taken into account by a single percentage figure. We propose the complexity of a text as an additional qualifying parameter to put the percentage score into the right perspective.

One simple measure is calculated as the average number of tags per word, i.e. the sum of all possible tags for all words, divided by the number of words. A text with a resulting score of 1.0 is therefore trivial to tag, as each word only has one possible tag. A better measure would be to disregard punctuation as this is almost always assigned a unique tag. An even better measure would consider only the ambiguous words, that is dropping any item (word or punctuation) that is uniquely labeled by the look-up (or guessing) procedure. For instance, the 3 scores for the texts used in the experiment reported here are shown in the table below:

Score	SM	NPM	AM
1984	1.55	1.60	2.49
Republic	1.63	1.72	2.37

Figure 7: Different measures of text ambiguity

SM (Simple measure) = number of tags/number of tokens
NPM (Non-Punctuation Measure) = number of non-punctuation tags/ number of non-punctuation tokens

AM (Ambiguity Measure) = number of tags assigned to ambiguous tokens/ number of ambiguous tokens.
The complexity of a text to be tagged is strongly dependent on the tagset used in the tagger: Q= Q1*Q2, where:
Q1 is one of SM, NPM or AM above and
$Q2 = \Sigma(1-p_{MLi})/(\varepsilon+N_{token})$
N_{token} is the number of tagged tokens in a text. Depending on the measure adopted as Q1, this number would be the number of all tokens, the number of non-punctuation tokens or the number of ambiguous tokens.
p_{MLi} is the most probable tag of $word_i$ (lexical probability)
ε is a small, non-zero, constant.

This measure for the complexity of a text would assign a zero value for texts with no ambiguous items. For unambiguous words, their contribution in the summation would be 0, while very probable tags would have a very small contribution. The metrics above has also the advantage that ponders the average ambiguity. This is relevant in those texts/languages which have few highly ambiguous words (say 10 different tags) but a lot of unambiguous words, which may be contrasted with texts/languages where most words are ambiguous (3-4 different tags). The first case should be easier for the tagger and the proposed metrics takes it into account. In the table above, one can see that the AM measure is higher for “1984” than for

"The Republic" (SM and NPM are lower). This was because more ambiguous words were used in Plato's text and although the average number of tags for each ambiguous words is smaller than in the case of "1984", the difficulty of "The Republic" excerpt was corrected by the Q2 term in the evaluation of Q. With any of the three measures adopted for Q1 above, the complexity of the test texts extracted from "The Republic" was higher than the texts extracted from "1984". This difference in complexity explains why the tagging results are better for the latter.

The text complexity is a useful parameter for estimating the accuracy with which the given text is expected to be tagged. For an "easy" text, one can use simple taggers, but for "difficult" texts a "more advanced" tagger should be necessary. We define the IQ of a tagger based on the number of cases when it made a non-trivial choice:

$$IQ = 100*\Sigma(p_{REAL\_i} - p_{ML\_i})/N_{token}$$

where p_{ML_i} and N_{token} are the same as before, and p_{REAL_i} is the lexical probability of the really selected tag in case of $token_i$. As one can notice, the tagger is not given credit for the unambiguous words. The same goes for those cases where the selected tag is the most likely one. The normalisation is necessary to cope with texts of different length used in different experiments.

Conclusions

It has been shown that it is quite easy to adapt a language dependent probabilistic tagger to work with data from other languages as well. Due to the way the resource files are created the training process is extremely fast. A client/server architecture provides an ideal framework for programs that require large linguistic resources, and the client implementation in Java means that it is possible to run the tagger on a wide variety of platforms. With a comparatively small amount of training data it is possible to reach quite impressive results.

The empirical methodology we described for deriving a convenient tagset (i.e. informative enough, manageable for the tagger and recoverable for a tiered tagging approach) from a large set of morpho-syntactic description codes proved to be successful. An old misconception, namely that highly inflected languages are doomed to poor performances when processed with probabilistic methods has been shown to be completely wrong. In fact, we do believe that a highly inflected language has better chances for being tagged accurately that other languages. The motivation is based on the fact that inflected words are less ambiguous than the base forms and in many cases they are simply unambiguous. Therefore, unambiguous words would act as tagging islands/clues for the rest of the words in the text. This way the number of possibilities to be considered in finding the most probable tags assignment is significantly reduced.

We proposed a measure for the complexity of a text to be tagged and an IQ figure a tagger. The complexity of texts varies not only across the languages but more often than one would expect within the same language. This complexity value might influence the choice of tagging engine to be used for disambiguating.

Acknowledgments

The work reported here was initiated as a joint experiment between Romanian Academy and Birmingham University within the TELRI Copernicus Concerted Action (Mason & Tufis, 1998), based on language resources developed under Copernicus Joint Project "MULTEXT-East" (Orwell's "1984") and TELRI Copernicus Concerted Action (Plato's "The Republic"). Given the very promising results, the continuation of the work was further granted by the Romanian Ministry of Science and Technology.

References

Abney, S.(1997): Part-of-Speech Tagging and Partial Parsing. In Young, S., Bloothooft, G. (eds.) *Corpus Based Methods in Language and Speech Processing* (pp. 118-136) Text, Speech and Language Technology Series, Kluwer Academic Publishers

Baayen, H., Sproat, R. (1996): Estimating Lexical Priors for Low-Frequency Morphologically ambiguous Forms in *Computational Linguistics*, vol. 22, no. 2 (pp. 155-166), June 1996

Berger, A., L., Della Pietra, S., A., Della Pietra, V., J. (1996): A Maximum Entropy Approach to Natural Language Processing in *Computational Linguistics*, vol. 22, no. 1 (pp. 39-72), March 1996

Elworthy, D. (1995): Tagset Design and Inflected Languages, *Proceedings of the ACL SIGDAT Workshop*, Dublin, (also available as cmp-lg archive 9504002)

Mason, O., Tufis, D. (1998, to appear): Probabilistic Tagging in a Multi-lingual Environment: Making an English Tagger Understand Romanian *Proceedings of the Third International TELRI Seminar*, Montecatini, October 1997

Tufis, D. (1998): Tiered Tagging. In *International Journal on Information Science and Technology*, vol. 1, no. 2, Editura Academiei, Bucharest, 1998

Tufis, D. (1997): A Generic Platform for Developing Language Resources and Applications, in *Proceedings of the Second International TELRI Seminar* (pp. 165-184) Kaunas, April 1997

Tufis, D., Barbu, A.M., Patrascu, V., Rotariu, G., Popescu, C. (1997):Corpora and Corpus-Based Morpho-Lexical Processing in Tufis D., Andersen P.(eds.): *Recent Advances in Romanian Language Technology*, Editura Academiei (pp. 35-56), Bucharest, 1997

Tzoukermann, E., Radev, D. (1997): Tagging French Without Lexical Probabilities - Combining Linguistic Knowledge and Statistical Learning *cmp-lg/9/10002*, 10 October, 1997

Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L., Palmucci, J. (1993): Coping with Ambiguity and Unknown Words through Probabilistic Models in *Computational Linguistics*, vol. 19, no. 2 (pp. 219-242), June 1993