Probabilistic Tagging in a Multi-lingual Environment: Making an English Tagger Understand Romanian[1]

Oliver Mason
Corpus Research
The University of Birmingham
United Kingdom
O.Mason@bham.ac.uk

Dan Tufiş
Institute for Research in Informatics
Centre for Artificial Intelligence
Romanian Academy,Bucharest
Romania
tufis@valhalla.racai.ro

August 1997

[1]Proc. of the 3-rd TELRI European Conference, Montecatini, Italy, Oct.1997

1 Introduction

This document describes a tagging experiment where a probabilistic tagger developed for English has been made language independent in order to work on Romanian data. The results of the tagging are very encouraging, with an error rate below 3%. In order to make those statistics more meaningful, a metric is proposed which allows more accurate comparisons between tagging results. The software described in this article is freely available.

2 Principles of P-O-S Tagging

There are two main approaches to part-of-speech tagging: rule-based and probabilistic. The tagger presented in this document belongs to the purely probabilistic ones. That means, that for disambiguating tags within a text it only uses probabilities, and no rule-based mechanism.

The first step in any tagging process is to look up the token to be tagged 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 possible to 'hand-code' a probabilistic language model. These models are generally created from training data, ie they learn by example. This is the way QTAG works.

The tagger used for the experiment presented here was QTAG, a probabilistic tagger which had previously been used at Corpus Research in combination with an email interface[1] It was re-implemented in order to make it language independent.

The basic algorithm is fairly straightforward: at first, the tagger looks up all possible tags that the current word can have, together with their respective probabilities. This information (which holds for the word in isolation) is then combined with the probability for each tag to occur in a sequence preceeded 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 words are evaluated.

3 The Algorithm

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.

[1] See `http:www-clg.bham.ac.uk/tagger.html`

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 $P_w = P(tag|token)$, the probability of the token to have the specified tag
   (b) calculate $P_c = P(tag|t_1, t_2)$, the probability of the tag to follow the tags t_1 and t_2.
   (c) calculate $P_{w,c} = P_w \times P_c$, the joint probability of the individual tag assignment together with the contextual probability.
5. repeat the computation for the other two tags, 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 internally. 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[2].

4 Client/Server

The tagger is implemented in a client-server model. The knowledge base resides on the server, and the tagging engine establishes a network connection to the server and retrieves the relevant data for tags and tag sequence probabilities from it. This has several advantages:

1. The resource data is in a well defined place and can be shared between different invocations of the tagger
2. The workload can be distributed across different machines
3. The memory usage of the tagger decreases enormously
4. The tagger can be used on small machines
5. The server can easily be used by other applications as well

The server is implemented in C, while the client is written in Java. That way it can run on all kinds of machines, provided there is a server available.

[2]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. Thus, only tag assignments with low confidence might have to be checked by a human for verification

5 Adapting the Tagger

In order to enable the tagger to work with Romanian data, all elements specific to English that had been hard-coded had to be removed. This mainly concerned the inbuilt morphologic component that would deal with unknown words, ie those that were not contained in the lexicon. All other information is contained in external resource files which are not specific to a language with regards to their format.

A separate issue concerns dealing with non-ASCII characters. The way which was chosen for those was to use their SGML entity names, a literal description enclosed in an ampersand and a semicolon, eg `ä`. As the input data for tagger is supposed to be tokenised (ie tokens separated by white spaces) the impact of an extended character on segmentation was not considered.

To increase the flexibility of the tagger for future uses with further languages the internal lexicon and context information were taken out of the processing core and the tagger was re-implemented following a client-server model: the client requests all necessary data from the server through a special protocol, and this allows to switch to a different tagset (ie a different language) easily.

The hard-coded morphology was replaced by another resource, a list of the final three letters of all words from the lexicon with their respective tag probabilities. This mechanism seems sufficiently general to yield good results even in languages with a rather complex morphology at little cost.

As the tagging program had to be changed anyway, it was re-implemented in Java, in order to allow the client to run on different machines more easily. The server component is still written in C. The total size of the three relevant (executable) byte-code files is less than ten kilobytes.

Furthermore, since the tagger is implemented as a class, it can easily be used as a module in other linguistic applications.

6 Language Resources

The Romanian word-forms lexicon was created based on a 35.000-lemma lexicon by means of our EGLU natural language processing platform (D. Tufiş: "A Generic Platform for Developing Language Resources and Applications", in Proceeding of the Second International TELRI Seminar, Kaunas, April 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 Romanian word-form lexicon is actually made of two parts: the main one contains only words attested by the Explanatory Dictionary of Romanian (DEX); all the other words, appearing in the corpus were entered an auxiliary lexicon.

The auxiliary lexicon contains, among other things, proper names, technical terms and the weird (made-up) words from Orwells *1984* (newspeak dialect).

The table below provides information on the data content of the main dictionary that is used for the corpus analysis.

Entries	Lemmas	MSDs	AMB-MSD
439827	33030	661	1326

The MSDs (Morpho Syntactic Descriptions) represent a set of codes as developed in the MULTEXT-EAST project (Tomaž Erjavec, Monica Monachini (eds)"MULTEXT-EAST: Specifications and Notation for Lexicon Encoding", WP1, Task 1.1 final report, September 1997). 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. That is, the positions in a string of characters are numbered 0, 1, 2, etc., and are used in the following way:

- the character at position 0 encodes part-of-speech;
- each character at position 1, 2,...,n, encodes the value of one attribute (person, gender, number, etc.), using the one-character code;
- if an attribute does not apply, the corresponding position in the string contains the special marker - (hyphen).

By convention, trailing hyphens are not included in the lexical MSDs. Such specifications provide a simple and relatively compact encoding, and are in intention similar to feature-structure encoding used in unification-based grammar formalisms.

The attributes and mainly the values of the attributes were chosen considering only word-level encoding. This is why, some values described in grammar text-books, but assuming compounding (such as compound tenses), were not considered in the MSD encoding.

For a given word-form several MSDs might be applicable (accounting this way for homography). The set of all the MSDs applicable to a given word defines the MSD-ambiguity class for that word. The Romanian lexicon contains 1326 MSD-ambiguity classes (for more details on the Romanian dictionary and corpora encoding and several relevant statistics see: D.Tufiş, et all "Corpora and Corpus-Based Morpho-Lexical Processing" in Tufiş D., Andersen P. (eds.): *Recent Advances in Romanian Language Technology*, Editura Academiei, Bucharest, 1997).

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 *Republic.* A brief overview of these texts is given below:

Sub-Corpus	Occurrences	Wordforms	Lemmas	MSDs	AMB-MSD
1984	101460	14037	7019	444	529
Republic	114720	10350	4697	400	498
Common	56804	4016	2527	344	402

The existent SGML mark-up (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” 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.

```
&Icirc;ntr-  Spsay
o Tifsr
zi Ncfsrn
senin&abreve; Afpfsrn
&scedil;i Ccssp
friguroas&abreve; Afpfsrn
de Spsa
aprilie Ncms-n
,
pe Spsa
c&acirc;nd Rw
ceasurile Ncfpry
b&abreve;teau Vmii3p
ora Ncfsry
treisprezece Mc-p-l
,
Winston Npms-n
Smith Np
,
cu Spsa
b&abreve;rbia Ncfsry
&icirc;nfundat&abreve; Afpfsrn
&icirc;n Spsa
piept Ncms-n
pentru Spsa
a Qn
sc&abreve;pa Vmnp
de Spsa
v&acirc;ntul Ncmsry
care Dw3--r---e
-l Pp3msa--y-----w
lua Vmii3s
pe Spsa
sus Rgp
,
se Px3--a--------w
strecur&abreve; Vmis3s
iute Rgp
prin Spsa
u&scedil;ile Ncfpry
de Spsa
```

```
sticl&abreve; Ncfsrn
...
```

6.1 The Tagset

The tagset for Romanian contains 79 tags for different morpho-syntactic categories, plus 8 tags for punctuation. The tagset has been derived by a trial-error procedure from the 674 morphosyntactic description codes (MSDs) defined for encoding the Romanian lexicon. By analysing the MSD clustering in the ambiguity classes, we started to eliminate several attributes in the MSD codes that would reduce 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.

The generalisation process was performed according to the following general guide lines:

- preserve the attribute encoded at position 0 (POS);
- eliminate the attributes that can be unambiguosly 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 othographic form of the words in the given POS class;
- preserve the attributes which correlates in co-occuring words.

Let us consider a few examples to clarify this methodological procedure. In case of nouns and adjectives the attributes 'case' and 'number' were preserved since when the latter modifies the former, their case and number must agree. Since nouns and adjectives are ambiguous in many cases with respect to these attributes when considered in isolation, but rarely when considered in co-occurrence, 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 word-form itself, and in those very rare cases when it is not, the immediate context (plus the agreement rule) does the job. We have not found in our corpora even one instance where this would not hold. Although it was preserved in the initial tagset, when it was removed the accuracy of the tagger increased (the number of tags decreased significantly).

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 băiat* versus *frumosul băiatului* or *iubitul băiat* versus *iubitul băiatului* or *băiatul iubit*).

With verbs, the distinction was preserved between finite verbs (main and auxiliary) and non-finite verbs (infinitive, participle and gerund). For finite verbs the attribute 'person' was preserved and only for 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 problematic categories were pronoun and determiner (this class contains what grammar books traditionally call adjectival pronouns and due to several commonalities 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 criteria the tagset generator 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 this cases and are insensitive to number). The next step was to classify the pronominal and determiner MSDs according to their types, number and case. With these new criteria, the number of tags remain the same, but the clustering of MSDs was more linguistically motivated. Then, by observing the MSD ambiguity classes, we manually assigned to a few number of MSDs different tags, so that hard statistical disambiguation should be avoided. The most obvious case was the distinction between numeral and article readings for the words *un* (*a/ an* or *one* masculine) *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.

Similar considerations applied for all the other word classes. In the table below the actual tagset is given.

```
1.   A    = Adjective
2.   AN   = Adjective, indefinite
3.   APN  = Adjective, plural, indefinite
4.   APON = Adjective, plural, oblique,indefinite
5.   APOY = Adjective, plural, oblique, definite
6.   APRY = Adjective, plural, direct, definite
7.   ASN  = Adjective, singular, indefinite
8.   ASON = Adjective, singular, oblique, indefinite
9.   ASOY = Adjective, singular, oblique, definite
10.  ASRY = Adjective, singular, direct, definite
11.  ASVN = Adjective, singular, vocative, indefinite
12.  ASVY = Adjective, singular, vocative, definite
13.  C    = Conjunction
14.  CVR  = Conjunction or Adverb
15.  I    = Interjection
16.  M    = Numeral
17.  NP   = Proper Noun
18.  NN   = Common Noun, singular
19.  NPN  = Common Noun, plural, indefinite
20.  NPOY = Common Noun, plural, oblique, definite
21.  NPRN = Common Noun, plural, direct, indefinite
22.  NPRY = Common Noun, plural, direct, definite
23.  NPVY = Common Noun, plural, vocative, definite
```

24. NSN = Common Noun, singular, indefinite
25. NSON = Common Noun, singular, oblique, indefinite
26. NSOY = Common Noun, singular, oblique, definite
27. NSRN = Common Noun, singular, direct, indefinite
28. NSRY = Common Noun, singular, direct, definite
29. NSVN = Common Noun, singular, vocative, indefinite
30. NSVY = Common Noun, singular, vocative, definite
31. NSY = Common Noun, singular, definite
32. PI = Quantifier Pronoun or Determiner (Indefinite or negative)
33. PXA = Reflexive Pronoun, accusative
34. PXD = Reflexive Pronoun, dative
35. PSP = Pronoun or Determiner, possessive or emphatic, plural
36. PSS = Pronoun or Determiner, possessive or emphatic, singular
37. PPPA = Personal Pronoun, plural, accusative, week form
38. PPPD = Personal Pronoun, plural, dative
39. PPSA = Personal Pronoun, singular, accusative
40. PPSD = Personal Pronoun, singular, dative
41. PPSN = Personal Pronoun, singular, nominative, non-third person
42. PPSO = Personal Pronoun, singular, oblique
43. PPSR = Personal Pronoun, singular, direct
44. PPPO = Personal Pronoun, plural, oblique
45. PPPR = Personal Pronoun, plural, direct
46. RELO = Pronoun or Determiner, relative, oblique
47. RELR = Pronoun or Determiner, relative, direct
48. DMPO = Pronoun or Determiner, demonstrative, plural, oblique
49. DMSO = Pronoun or Determiner, demonstrative, singular, oblique
50. DMPR = Pronoun or Determiner, demonstrative, plural, direct
51. DMSR = Pronoun or Determiner, demonstrative, singular, direct
52. QN = Infinival Particle
53. QS = Subjunctive Particle
54. QF = Future Particle
55. QZ = Negative Particle
56. R = Adverb
57. S = Preposition
58. TP = Article, indefinite or possessive, plural
59. TPO = Article, non-possessive, plural, oblique
60. TPR = Article, non-possessive, plural, direct
61. TS = Article, definite or possessive, singular
62. TSO = Article, non-possessive, singular, oblique
63. TSR = Article, non-possessive, singular, direct
64. V1 = Verb, main, 1st person
65. V2 = Verb, main, 2nd person
66. V3 = Verb, main, 3rd person
67. VA1 = Verb, auxiliary, 1st person
68. VA1P = Verb, auxiliary, 1st person, plural
69. VA1S = Verb, auxiliary, 1st person, singular
70. VA2P = Verb, auxiliary, 2nd person, plural
71. VA2S = Verb, auxiliary, 2nd person, singular
72. VA3 = Verb, auxiliary, 3rd person
73. VA3P = Verb, auxiliary, 3rd person, plural

```
74.  VA3S = Verb, auxiliary, 3rd person, singular
75.  VG   = Verb, gerund
76.  VN   = Verb, infinitive
77.  VP   = Verb, participle
78.  X    = Residual
79.  Y    = Abbreviation
```

6.2 The Training Corpus

The training corpus, or (see example below) was generated from the MSD-annotated tokenised corpus by substituting the MSDs with their corresponding corpus tags (described in the previous section). We called this kind of resource an MSD-corpus.

```
&Icirc;ntr-  S
o TSR
zi NSRN
senin&abreve; ASRN
&scedil;i CVR
friguroas&abreve; ASRN
de S
aprilie NSN
,COMMA
pe S
c&acirc;nd R
ceasurile NPRY
b&abreve;teau V3
ora NSRY
treisprezece M
,COMMA
Winston NSN
Smith N
,COMMA
cu S
b&abreve;rbia NSRY
&icirc;nfundat&abreve; ASRN
&icirc;n S
piept NSN
pentru S
a Qn
sc&abreve;pa VN
de Sp
v&acirc;ntul NSRY
care PR
-l PSA
lua V3
pe S
sus R
,COMMA
```

```
se PA
strecur&abreve; V3
iute R
prin S
u&scedil;ile NPRY
de S
sticl&abreve; NSRN
...
```

Out of the training corpus, 80% was retained for the proper training and the rest of 20% (the first parts of both *1984* and *Republic*) used for the validation purposes.

6.3 The Training Process

The training process is merely a reformatting 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.

The tagger is ready for use with the new resources as soon as the data files are accessible to the server program (it is not even necessary to restart the server for this). The output format of the tagger, which we call tag-corpus is a vertical text with all possible tags; the tags have a probability value assigned and are sorted so that the most likely tag comes first (see example below[3]).

```
...
o [TSR:-4.7998][PSA:-7.1013][Qf:-9.839][VA3S:-11.4813][I:-11.8867]
zi [NSRN:-1.0662][V2:-10.6697]
...
```

This allows expressing a judgement 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, the decision is more certain than 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.

7 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. The complexity of a text is proposed as an additional qualifying parameter to put the percentage score into the right perspective.

[3]The probabilities are internally represented by logarithmic values in order to avoid problems that would arise during calculations with very small numerical values.

One simple measure is calculated as the average number of tags per word, ie 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 sample score for an English text is 1.92, while the sample Romanian scores are 1.69 (Orwell's *1984*) and 1.72 (Plato's *Republic*). These scores were computed by ignoring the punctuation. The larger the score, the higher the ambiguity of the text and the more difficult it is to tag. The complexity of a text to be tagged is mostly dependent on the tagset used in the tagger.

In evaluating the complexity of the Romanian texts we should note that it was decreased due to some encoding decisions which are largely described in D. Tufiş et all. "Corpora and Corpus-Based Morpho-Lexical Processing". For instance by considering the case syncretism in Romanian, the Nominative and Accusative were conflated into Direct case while Genitive and Dative were conflated into the Oblique case. For instance, considering the tags for Nouns, Adjectives, Pronouns, Determiners and Articles for which the Case information has been preserved (as oposed to Numerals, Prepositions and Abbreviations where this distinction has beed dropped), one could find a large difference in the complexity of the same text when case syncretism is expanded as versus the case when sycretic cases are conflated. In Plato's *Republic* the number of Case relevant tags, which the tagger would be supposed to differentiate would be in the first case 138585 while in the second one this number is just 24425.

Obviouslly the cardinality of the tagset when expanding the Case syncretism would be much larger (instead of the actual 79 codes, the tagset would contain almost 500 codes).

Another important decision was to eliminate the neuter gender since in singular all the neuters behave like masculines and in plural like feminines. With only these two encoding decisions, the complexity of the texts decreased significantly down to the before-mentioned manageable values.

8 Evaluation

The tagger was evaluated twice on about 23.000 words, the beginning passages of *1984* and*Republic*. They had not been included in the training data. A third test included tagging the full training corpus with resources taken from the total amount of data. The results are highly encouraging, with correct assignments between 97% and 98%.

As shown in the previous sections, the tagger output is a a tabular format with each word on a line followed by an ordered list of pairs [tag:probability]". The above mentioned accuracy was measured considered only the highest probability assigned tag.

Another option is 2-best tagging (a variant of K-best tagging, (see Brill, Computational Linguistics, 1995), that is to preserve in the output the first two best rated tags, when the difference among their probabilities is below a specific threshold. We set this threshold to 1, and out of the 641 errors in the first 23643 words in *1984*, 126 were removed making an increase of accuracy from 97.288% to 97.821% at the price of increasing the average number of tags per word from 1 to 1.04, that is out of the 23643 tagged words, only 969 had 2 tags.

By analysing the 3085 tagging errors in the entire Orwell's *1984* we discovered that 189 errors attributed to the tagger were human made errors in the hand-tagged training corpus. The tags atributed by the tagger were in all these cases the correct ones, so the actual accuracy for this experiment even better than the one reported above. Several other (real errors) errors suggested further modifications on the tagset. Some of these modifications are discussed below.

We run the tagger on several tagsets and in those cases where it was definitely wrong (that is the propability of the chosen tag was much larger than the probability of the correct one) we modified by hand the MSD and the corresponding tag. This happened for instance with the wordforms *şi* and *şi-* which in the lexicon are listed as both conjunction and adverb. Being very hard to be classified correctly, the two interpretations were merged into one tag (CVR). Another example is given by the word *fi*, initially marked as an aspectual particle, but which rarely has been correctly identified by the tagger. Since this is a very frequent word in Romanian, it's "contribution" to the errors list was significant(4.28% of all the errors). We turned it into an infinitive (what most grammar books would do), and all these errors have been prunned. Other follow-up modification of the tagset consisted in making distinction among the particles (Qn, Qs, Qf) initially conflated into one tag (Q). This distiction eliminated several disambiguation errors (such as distinction between auxiliary and main usage of some verbs). On the same line, must be mentioned making a distiction (which in the initial tagset was not done) among the proper nouns and common nouns. Making this distinction helped in diminishing tagging errors of the word *lui* which when precedes a proper noun is always a genitival article (not a pronoun - 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 prefered to the more general tag ASN (adjectives, singular, indefinite). Again, by observing the MSDs mapped into the ASN tag, we noticed only masculine, indefinite adjectives in singular, which are Case undetermined. Therefore, a natural decision was to conflate the ASRN and ASN tags. This decision eliminated tagging errors for those few adjectives (but very commonly used) which in singular and indefinite forms have the direct Case of the feminine foms identical to the masculine forms (for instance "mare"-big).

9 Conclusion and Further Work

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 takes hardly any time at all. A client/server architecture provides an ideal framework for programs that require linguistic data, 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 and manageable for the tagger) 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 a high tagging accuracy 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 signigicantly reduced.

We plan further experiments with several other tagsets and we are currently investigating the possibility to develop a postprocessor (essentially a dictionary lookup and string pattern matching program) for the tagger, able to recover the MSD-corpus from a given tag-corpus.