Master's Thesis in Computer Science: Machine learning-based classifiers in the Direkt Profil grammatical profiling system

Jonas Thulin, F00 [1]

Department of Computer Science,
Lund Institute of Technology,
Lund University, Lund, Sweden

11th January 2007

[1]Supervisors: Jonas Granfeldt and Pierre Nugues

Abstract

Direkt Profil (DP) is a system for grammatical profiling. It detects, annotates and displays grammatical constructs, both correct and incorrect, in freely-written texts by Swedish-speaking learners of French. It can also determine the learner's developmental stage, given a text with enough identifying attributes.

The scope of my work is the final step, the classification of the text as being typical for a certain stage, for which machine learning (ML) methods, more specifically C4.5, LMT (Logistic Model Tree) and SVM (Support Vector Machine), have been applied.

This thesis is part of a longer-term research project, led by Jonas Granfeldt and Suzanne Schlyter at the Centre for languages and literature at Lund University. The research project aims at increasing our knowledge regarding how Swedish-speaking learners acquire proficiency in written French. During a three-year period, commencing in 2005, it is being financed by the Swedish Science Council.

In my experiments with an early version (1.5.2) of the profiling system, precision and recall values of a ternary classifier (basic/intermediate/native), based on support vector machines, have reached 70–83 %. The system has also been tested with C4.5- and logistic model tree-based classifiers, yielding similar (LMT) or slightly inferior (C4.5) results.

Direkt Profil 2.0 gives similar performance even for a quintary classifier, and ternary classifier precision and recall is somewhat improved as well (up to 88 %). The Naive Bayes method yields a small further overall precision/recall increase, and is much faster than SMO (SVM) on the experiment corpus.

This project paves the way for further experiments with parameter selection and classifier performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning is about using computers not only for calculations and data retrieval, but combining those two capabilities of a computer system to make it appear to be learning and making rational decisions according to previously observed circumstances and previous actions or reactions, and not only act according to a fixed plan. This is especially important when the function to be performed by the system is too complicated to be described in code, or perhaps the condition-action mappings are unknown.

It has been used not only for search engines, medical diagnosis, and speech and handwriting recognition, but also for such applications as stock market analysis, classifying DNA sequences, speech and handwriting recognition, and robot locomotion.

Machine learning technology is usable in a vast area of applications, and more uses are being found out as time passes. It allows computer systems to improve in dynamic environments, where the input signals are unknown, and the best decisions to be made can only be learnt from history. One such example is speech recognition, where the user's accent and tendencies to sometimes slur words impedes accuracy, when a new speech recognition system has been installed, but as the user corrects the mistakes and trains the system further on specific problem words and phrases, the system makes fewer and fewer mistakes.

Chapter 2

Learning a Foreign Language

2.1 Introduction

In today's global society, the importance of knowing several languages is greater than ever before. Hence, there is a large interest in language learning and learners' application of their newly acquired skills, not only for the reason of curiosity, but also because more extensive knowledge about the learning process allows us to optimise it, so that one can learn a new language, in this case French, faster and more efficiently, by focusing on one's weak areas.

When learning a new language, there are certain milestones that everyone passes, some necessarily being in a specific order. To give an example, learners tend to use the present tense before the future, and construct simple sentences before being able to link clauses together into the complex structures typically used by more advanced users of the language (Schlyter, 2003). This does not only apply to non-native learners, but also to children who are native speakers (Clahsen, 1986).

However, some differences persist, most importantly that young children do not know how to write, but instead have the ability to build up a complete innate grammar with a simple formalism and memorised exceptions. A particularly interesting feature of non-native language is that non-native speakers, unless very advanced, more or less frequently pick the wrong gender on nouns and appear to have difficulties in making the inflected forms agree with each other in a phrase. Gender-number – and, though not in the case of French, case – agreement requires an immense amount of practice to be mastered well and fluently, as does fitting sentences together into longer units (coherence) and predicting the next few words that will appear.

2.2 The Common European Framework

An example with qualitative milestones is the Common European Framework of Reference for Languages (CEF; Council of Europe (2001)), with six developmental stages for non-native speakers, characterised in several ways. In table 2.1 these six stages are described in very broad and general terms. The entire CEF report, which is 264 pages long, contains further stage descriptors for more specific situations (studying, working, tourism etc.), as well as for specific aspects of language usage (grammatical accuracy, verbal fluency, adaptivity to the audience etc.).

You can think of the CEF stages as profiles that can be defined according to a number of attributes in the CEF. These attributes are expressed on the level of pragmatics; they state what the learner 'can do' with the foreign language at different stages of development.

The aim of Direkt Profil is to build and define grammatical profiles. The attributes are expressed mainly on the levels of morphology and syntax, but some more generic attributes refer to the lexicon.

So, in conclusion, the CEF is a well-established example of a *criterion-referenced* framework for defining language proficiency and development. Criterion-referenced here means that the learner's proficiency level is determined only by his/her own ability compared to a pre-defined set of criteria, and not to other learners. This is how the CEF levels, and also grades presently given in Swedish schools, are determined. As a contrast, the 1–5 school grading scale used in Swedish elementary and upper secondary schools until 1997 and 1996, respectively, were based on a bell curve resemblant of a normal distribution, and students got grades according to their national percentile rank, rather than the absolute proficiency level they had reached.

Direkt Profil has, just like the CEF and the current Swedish school grades, the aim to describe the learner's level on an absolute scale, where goals can be ticked off as they have been reached. However, DP presently deals more with grammar than with expressive ability, verbal fluency, protocol knowledge etc. Other important differences are that DP isolates a chosen set of characteristic features; currently[1] 111 features are counted; and that it takes a quantitative and mathematical approach to the classifying problem, rather than the qualitative one used in the CEF[2].

[1] Version 2.0

[2] The criteria in the table below all describe qualitative measures of language proficiency.

C	**Proficient User**
C2	Can understand with ease virtually everything heard or read. Can summarise information from different spoken and written sources, reconstructing arguments and accounts in a coherent presentation. Can express him/herself spontaneously, very fluently and precisely, differentiating finer shades of meaning even in more complex situations.
C1	Can understand a wide range of demanding, longer texts, and recognise implicit meaning. Can express him/herself fluently and spontaneously without much obvious searching for expressions. Can use language flexibly and effectively for social, academic and professional purposes. Can produce clear, well-structured, detailed text on complex subjects, showing controlled use of organizational patterns, connectors and cohesive devices.
B	**Independent User**
B2	Can understand the main ideas of complex text on both concrete and abstract topics, including technical discussions in his/her field of specialisation. Can interact with a degree of fluency and spontaneity that makes regular interaction with native speakers quite possible without strain for either party. Can produce clear, detailed text on a wide range of subjects and explain a viewpoint on a topical issue giving the advantages and disadvantages of various options.
B1	Can understand the main points of clear standard input on familiar matters regularly encountered in work, school, leisure, etc. Can deal with most situations likely to arise whilst travelling in an area where the language is spoken. Can produce simple connected text on topics which are familiar or of personal interest. Can describe experiences and events, dreams, hopes and ambitions and briefly give reasons and explanations for opinions and plans.
A	**Basic User**
A2	Can understand sentences and frequently used expressions related to areas of most immediate relevance (e.g. very basic personal and family information, shopping, local geography, employment). Can communicate in simple and routine tasks requiring a simple and direct exchange of information on familiar and routine matters. Can describe in simple terms aspects of his/her background, immediate environment and matters in areas of immediate need.
A1	Can understand and use familiar everyday expressions and very basic phrases aimed at the satisfaction of needs of a concrete type. Can introduce him/herself and others and can ask and answer questions about personal details such as where he/she lives, people he/she knows and things he/she has. Can interact in a simple way provided the other person talks slowly and clearly and is prepared to help.

Table 2.1: The six CEF levels (Council of Europe (2001), p. 24)

Chapter 3

Direkt Profil

3.1 Learner evaluation and profiling

Bartning and Schlyter (2004) have analysed interviews with Swedish-speaking French students, both from secondary schools and universities, and suggested a model for the acquisition of grammatically elaborate and correct spoken language. It is believed that there are similar progressions in written language acquisition, and in order to investigate this hypothesis further, secondary school learners have been given written tasks, to which the answers have been manually annotated and classified (Granfeldt et al., 2005a). Other fundamental works for this project are Granfeldt (2003) and Ågren (2005).

3.2 Data collection

Ågren (2005) gave Swedish upper secondary school students of French the task to write texts in French, among other things introducing themselves and their family and describing a picture series about two Swedish women going on vacation to Italy. The time allowance was sufficient to make automatised linguistic knowledge, not typing speed or groping for the right words, the most significant performance factor. No aids were permitted.

This corpus, CEFLE[1], contains 400 learner texts, and has been manually classified and partially (25 texts) annotated to be used as a benchmark for Direkt Profil's parsing and classification routines (Granfeldt et al., 2006). One elementary-level learner text is shown on the next page. It is displayed once again in Direkt Profil's user interface in figure 3.2.

The texts analysed in this project are those describing the trip to Italy, since it is the text to which the most attention has been given in the manual annotation and classification process. The series is designed specifically to test agreement between

[1] Corpus Écrit de Français Langue Étrangère

person/noun, (grammatical) gender and number in verb and noun phrases, but naturally spelling and lexis matter to a certain degree, as well as some aspectual phenomena, e.g. *imparfait* vs. *passé composé*. [2] To describe what happens in the series, one must pay attention to the gender of the nouns; many of them are feminine; as well as the main persons, when inflecting adjectives and participles.

[2] In French, the imperfect is used to indicate habits or events taking place over an extended time in the past, while the composed past is used for distinct events. Hence, the sentence 'I *was watching* a film when the telephone *rang*.' is in French 'Je *regardais* un film quand le téléphone *a sonné*.' The watching is the extended event and the ringing is the distinct, interrupting event.

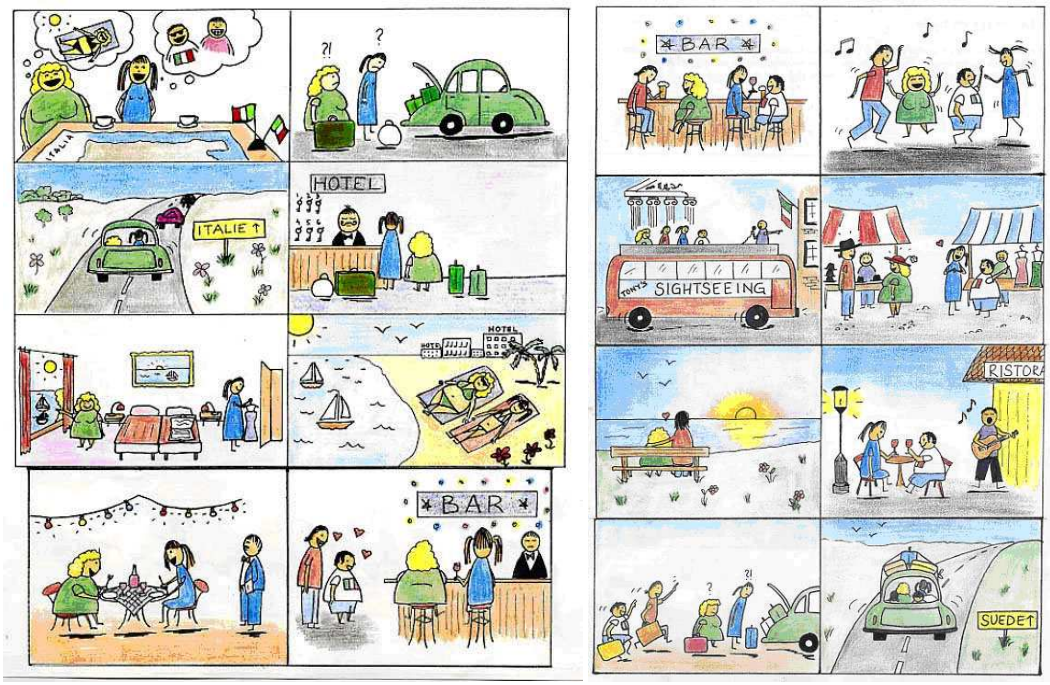

Figure 3.1: Trip to Italy picture series (Granfeldt et al., 2005b)

Example of a learner text ('Bodil') to the picture series above:

'Ellen et Marina sont a la maison de Ellen. Elles vont aller a Italie, en vacanse. Ellen pense a les garcons dans Italie, ils sont très jolis, et Marina pense a le soleil. A mecredi, c'est le jour, Marina et Ellen conduire la voiture verte en Italie!

A l'hotel il y a un très triste homme a la reception. Il reponse le questions, qui Marina pose. Les deux filles marchent a la chambre, numero trois. Elles ont un view fantastic, et elles peuvent voir le bateaus a la mer.

Le soleil brille, il fait chaud et beau, parce que elles vont a la beach. Dans l'eau deux persones faire de la voiles. A la beach, il ya des fleurs, et des palmes.

Le soir est ici, est les amis vont a un restaurant. Marina met un robe verte, est Ellen une robe bleue. Il manger le entrecôte avec pommes frites et elles boient le vin. Beacoup trop vin.

Puis, elles vont a un bar, et elles order le plus vin a un bartender. Deux garcons voir les belles filles! Måns a un chandail rouge et des pantalons bleus. Tor a un T-shirt avec les drapeu italian. Ils sont très jolie, pensent Marina et Ellen. Marina aime Måns, et Tor aime Ellen, ils dancer tout le soir. Ils ont très drôle.

Le prochain jour, le quatre amis vont a sightseeing, Tony's sightseeing. Ils vont a un market, et Marina et Måns achetent les chapaues noir et rouge. Ellen est amoreuse de Tor! Il est intelligtent, mignon, et très sympa!

Cet soir, Marina et Måns voir le soleil. Ellen et Tor boient le vin a un ristorante, et un homme jouer un instrument et il chanter. Très romantic.

Maintenant, le vacanse de Ellen et Marina et terminé. Elles sont très tristes, mais soudain, Tor et Måns apparant, et ils parlent, ils peuvent aller en Suede! Ellen, Marina, Tor et Måns sont ensemble! Quel vacanse! Amour, soleil, bon norriture, beacoup de vin et plus Amour. Ellen + Tor et Marina + Måns = VRAI!!'

Ph.	**Stages**	**1**	**2**	**3**	**4**	**5**	**6**
A.	% of sentences containing a verb (in a conversation)	20–40%	30–40%	50%	60%	70%	75%
B.	% of lexical verbs showing +/-finite opposition (types)	No opp.; % in finite forms 1–3sg	10–20% of types in opposition	About 50% in opposition	Most in opposition	All in opposition	+
C.	% of finite forms of lexical verbs in obligatory contexts (occurrences)	Finite forms 50%–75%	Finite forms 70–80%	Finite forms: 80–90%	Finite forms: 90–98%	Finite forms: 100%	+
D.	1^{st}, 2^{nd}, 3^{rd} pers. sing. (copula/aux) *est, a, va*	No opposition: *J'ai/ c'est*	Opposition *j'ai – il a* *je suis – il est*	Isolated errors **je va, *je a*	+	+	+
E.	% of 1^{st} pers. plural S-V agreement *nous* V-*ons* (occurrences)	–	70–80%	80–95%	Errors in complex constructions	+	+
F.	3^{rd} pers. plural S-V agreement with *viennent, veulent, prennent*	–	– *ils *prend*	Isolated cases of agreement	50% of cases with agreement	Some problems remain	+
G.	Object pronouns (placement)	–	SVO	S(v)oV	SovV appears	Productive	+ *(y, en)*
H.	% of gender agreement Article-Noun (occurrences)	55–75%	60–80%	65–85%	70–90%	75–95%	90–100%

Table 3.1: Developmental sequences adapted from Schlyter (2003); Bartning and Schlyter (2004). Legend: – = no occurrences; + = acquired at a native-like level; aux = auxiliary; pers. = person; S-V = Subject-Verb

3.3 Profile analysis

Profile analysis means collecting and analysing characteristic data from a given input, e.g. a written text or a spoken dialogue. The characteristic data could be, for instance the percentage of nouns having the correct gender (*mon* voiture or *ma* voiture?), which is phenomenon H in Table 3.1, or how frequently verbs are inflected correctly according to number ('La fille et le garçon *mange* dans un restaurant.'); something akin to phenomenon B in the same table. Apart from syntactic features, other interesting features are the occurrence of certain verb forms or words typically not used by students below a certain level. Version 2.0 of the Direkt Profil system finds out 111 features, which are transformed from counter values into percentages, and thus reduced in number to 58.

This large parameter space entails a need for automatic quantitative morphosyntactical and lexical analysis of freely written texts, since performing it manually is a laborious, tedious and error-prone task. However, natural language poses many difficulties for a computer, due to the fact that text analysis in a human-like way requires the processing capabilities and knowledge of a human. Not all human knowledge is encodable in a machine-friendly way, since the programmer or software designer cannot include all knowledge pertaining to all specific cases, especially those which are ambiguous in such a way that semantic knowledge is required. A good example of this would be the aspectual difference between the imperfect and the composed past mentioned above. Another problem would be a slightly misspelt word, where a human reader would easily realise what was meant, for instance in the sentence *'Ils ont fait des courses* sur un marche.' ('They went shopping on a step.') instead of *au marché* (in a market) The analyser currently believes that the student meant *sur une marche* (on a step), which makes no sense at all, even though it is the closest match (wrong gender, so there is no exact match).

3.4 History of the Direkt Profil system

Qualitative and quantitative linguistic progressions are clearly observable in written French, but manual profile analysis is too resource-demanding to be feasible on a larger scale. Therefore, a parsing system, Direkt Profil, has been developed[3].

The first prototype, implemented by (Kostadinov and Thulin, 2003) as the project part of a course in Language Processing and Computational Linguistics in the autumn of 2003, laid the foundation for the system up until now, but due to a shortage of time (allotment of only 160 man-hours), that prototype was non-operational. Thanks to Persson (2004), Kostadinov (2005), this project, as well as the other programmers' positions, the system has been made operational and

[3] A list of present and former team member is available online at `http://project.sol.lu.se/DirektProfil/medarbetare.html`.

refined further; when this report was finished, the most recent version was Direkt Profil 2.0.

Changes from the previous version (1.5.2) include:

- The system can detect and analyse clauses with nominal as well as pronominal subjects (in phrases with straight word order).
- Rudimentary spellchecking of verbs and nouns.

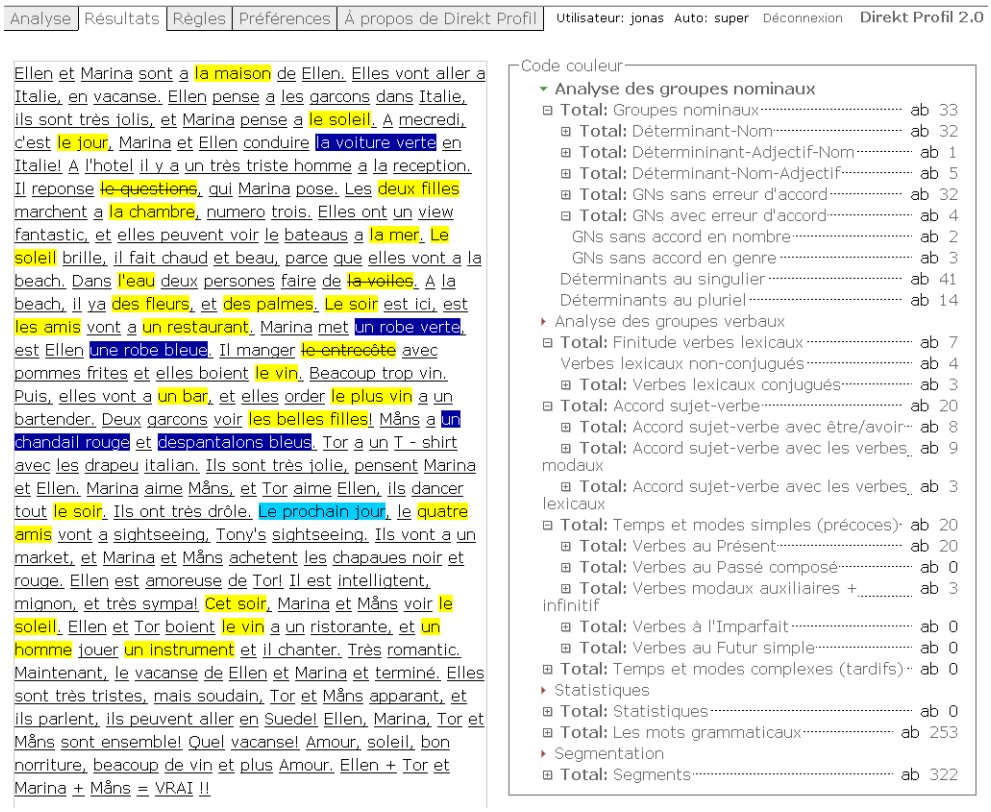

Figure 3.2: A learner text, visually marked up by Direkt Profil. Some grammatical attributes can be seen on the right side.

3.5 All counters in Direkt Profil 2.0

Here is the complete list of attributes (raw counter values) extracted by Direkt Profil 2.0:

c3000	Nominal phrases
c3200	Determiner-Noun
c3202	Determiner-Noun in agreement
c3201	Determiner-Noun not in agreement
c3300	Determiner-Adjective-Noun
c3302	Determiner-Adjective-Noun not in agreement
c3301	Determiner-Adjective-Noun in agreement
c3400	Determiner-Noun-Adjective
c3402	Determiner-Noun-Adjective with agreement
c3401	Determiner-Noun-Adjective without agreement
c3010	Nominal phrases without any agreement errors
c3012	Nominal phrases with number agreement
c3014	Nominal phrases with gender agreement
c3011	Nominal phrases with agreement errors
c3013	Nominal phrases without number agreement
c3015	Nominal phrases without gender agreement
c3020	Determiners in the singular
c3021	Determiners in the plural
c4000	Finiteness lexical verbs
c4100	Uninflected lexical verbs
c4200	Inflected lexical verbs
c4210	Without Subj-Verb agreement
c4220	With Subj-Verb agreement
c4300	Subject-Verb agreement
c4500	Subject-Verb agreement with être/avoir
c4501	With subject-verb agreement with être/avoir in the singular
c4502	Without subject-verb agreement with être/avoir in the singular
c4503	With subject-verb agreement with être/avoir in the 3rd pers. plural
c4504	Without subject-verb agreement with être/avoir in the 3rd pers. plural
c4505	With subject-verb agreement with être/avoir in the 1st pers. plural
c4506	Without subject-verb agreement with être/avoir in the 1st pers. plural
c4507	With subject-verb agreement with être/avoir in the 2nd pers. plural
c4508	Without subject-verb agreement with être/avoir in the 2nd pers. plural
c4600	Subject-Verb agreement with modal verbs
c4601	With subject-verb agreement with modal verbs in the singular
c4602	Without subject-verb agreement with modal verbs in the singular
c4603	With subject-verb agreement with modal verbs in the 3rd pers. plural
c4604	Without subject-verb agreement with modal verbs in the 3rd pers. plural
c4605	With subject-verb agreement with modal verbs in the 1st pers. plural
c4606	Without subject-verb agreement with modal verbs in the 1st pers. plural

c4607	With subject-verb agreement with modal verbs in the 2nd pers. plural
c4608	Without subject-verb agreement with modal verbs in the 2nd pers. plural
c4700	Subject-Verb agreement with lexical verbs
c4701	With subject-verb agreement with lexical verbs in the singular
c4702	Without subject-verb agreement with lexical verbs in the singular
c4703	With subject-verb agreement with lexical verbs in the 3rd pers. plural
c4704	Without subject-verb agreement with lexical verbs in the 3rd pers. plural
c4705	With subject-verb agreement with lexical verbs in the 1st pers. plural
c4706	Without subject-verb agreement with lexical verbs in the 1st pers. plural
c4707	With subject-verb agreement with lexical verbs in the 2nd pers. plural
c4708	Without subject-verb agreement with lexical verbs in the 2nd pers. plural
c5000	Simple (early) tenses and modes
c5100	Verbs in the present
c5115	Être / avoir in the present
c5135	Modal verbs in the present
c5155	Lexical verbs in the present
c5200	Verbs in the composed past
c5210	Without Subj-Verb agreement
c5220	With Subj-Verb agreement
c5300	Modal auxiliaries + infinitive
c5310	Without Subj-Verb agreement
c5320	With Subj-Verb agreement
c5400	Verbs in the imperfect
c5415	Être / avoir in the imperfect
c5435	Modal verbs in the imperfect
c5455	Lexical verbs in the imperfect
c5500	Verbs in the simple future
c5515	Être / avoir in the simple future
c5535	modal_futur_simple
c5555	lexical_futur_simple
c6000	Complex (late) tenses and modes
c6100	Verbs in the pluperfect
c6110	Verbs in the pluperfect without S-V agreement
c6120	Verbs in the pluperfect with S-V agreement
c6200	Verbs in the conditional
c6215	Être/Avoir in the conditional
c6235	Modal verbs in the conditional
c6255	Lexical verbs in the conditional
c6300	Être/Avoir in a different tense or mode
c9000	Statistics
c9001	Number of words
c9002	Number of sentences
c9003	Average sentence length

c9004	Unknown words
c9005	Formulaic expressions
c9006	number_of_hard_del[.!?;]
c9007	hidden_segments
c9008	Number of characters
c9009	Average word length
c0000	Grammatical words
c0001	Sentence delimiters
c0002	Conjunctions
c0003	Prepositions
c0004	Determiners
c0005	Nouns
c0006	Nominative pronouns
c0007	Verbs
c0008	Pending
c0010	Segment
c0011	pro_nom_segment
c0012	nom_segment
c0013	ver_segment
c0014	pre_segment
c0015	mwe_segment
c0016	con_segment
c0017	del_segment
c0018	unknown_segment
c0019	pro_nom_ver_segment
c0020	pro_nom_ver_trans_segment
c0021	name_segment
c0022	NP_ver_segment
c0023	NP_ver_trans_segment

In addition to these, the following ones have been added via in-house tools:

- Word count
- Unique word count
- TTR
- Percentage of the 1,000 most common words
- Percentage of the next 2,000 words
- Percentage of less frequent words

- Percentage of non-dictionary words
- Classification { 1, 2, 3, 4, 6}[4]

3.6 The Classifier Module

Once the TTR, lexical frequency, and grammatical features have been counted, the next step is adding a classifier module, which can, given those counter values, accurately predict the learner's developmental level of written French. This is accomplished using Machine Learning (ML) algorithms, which learn how to classify from a set of given, pre-classified examples. These algorithms are statistical, and can be inspired by such different things as information theory, neuroscience and separating hyperplanes. Virtually anything that can be coded and that works for fitting and applying a statistical model can be used.

[4]The numbers denote beginner, novice, intermediate or advanced learner (1–4), or native speaker (6).

Chapter 4

Machine Learning

Machine learning can be accomplished in a supervised or an unsupervised way. In supervised learning, the system receives a dataset with different example parameter values and decisions/classification, from which it infers a mathematical function, which automatically maps an input signal to an output signal. So, it figures out what it is supposed to do. This project uses only supervised methods.

Unsupervised learning, on the other hand, means that the system acts and observes the consequences of its actions, without referring to any predefined type cases other than those previously observed. This is pure 'learning by doing' or trial-and-error. Compared to supervised learning, unsupervised methods perform poorly in the beginning, when they are untuned, but as they tune themselves, performance increases. It can be argued that using unsupervised learning, a classifying system should be able to set up hypotheses that no human can figure out, due to their complexity. If unsupervised methods were used for this project, the machine learning system would have to find out the learner stage hypothesis all on its own, which would probably require much more training data than is available. One would also run the risk of obtaining a hypothesis too complex or specific to aid researchers (think of Occam's razor).

To evaluate classifier performance given by a machine learning scheme, either a special testing set or a cross validation technique may be employed. A test set contains pre-classified examples different to those in the training set, and is used only for evaluation, not for training. If data are scarce, it is sensible to use cross validation in order not to waste any data, which could be useful to enhance classifier performance; all data are used both for training the classifier and for testing its performance.

More examples does not necessarily mean better classifier performance. Even though the classifier becomes better on the training set it could actually perform worse on the test data. This is due to the over-fitting of the classifier transfer function, so that it fits too tightly to the training data and the border between classes is jagged rather than smooth, unlike how it usually should be.

4.1 Decision tree learning

A decision tree is a tree-formed set of rules which leads to a decision. Each rule may refer to another rule or to a final decision. For example, you could say that you like playing tennis, unless the weather is too uncomfortable, and illustrate the uncomfortable conditions in a tree, such as the one below.

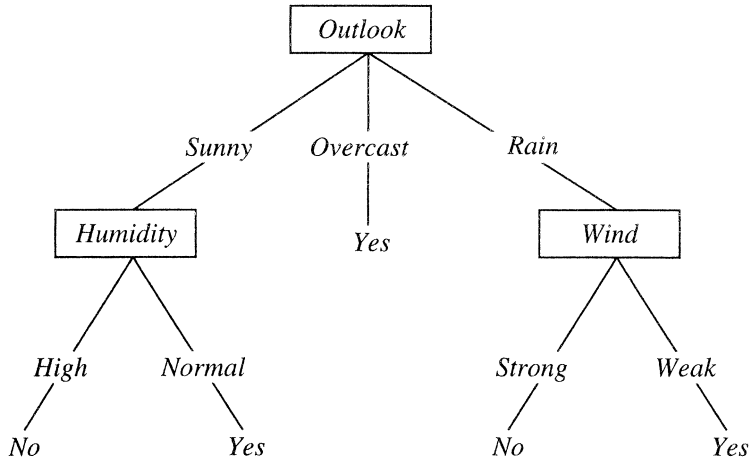

Figure 4.1: Example decision tree. After Quinlan (1986)

Using machine learning algorithms, a computer can infer such decision trees from tables with examples showing different conditions (attributes) and outcomes (classifications or decisions). It is possible to represent a decision tree as a table or as a set of Boolean expressions. For example, the tree above can be written as:

or

$$(Outlook = Sunny \wedge Humidity = Normal) \\ \vee (Outlook = Overcast) \\ \vee (Outlook = Rain \wedge Wind = Weak)$$

.

To construct a decision tree for classification by one or several nominal (discrete and finite) attributes, such as {1, 2, 3, 4} or {black, white}, one can use a recursive algorithm called ID3 (Iterative Dichotomiser 3). It is a greedy algorithm; in each step (tree node) it chooses the parameter which best predicts the outcome (has the highest information gain). Once a node has been processed, the algorithm tests the next best predictor, given the previously used ones. This recursive optimisation continues until all examples in the training set have been correctly classified, which most likely leads to overtraining and lower than optimal performance on the testing set.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 4.1: The table corresponding to the tree above. After (Quinlan, 1986)

A measure of the information value of the elements p_i in the set S is the entropy, which is defined as

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

There is also a measure of the information gain from using an attribute A, existing for the elements in the set S:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{S} Entropy(S_v)$$

C4.5 (Quinlan, 1993), adds to the ID3 method two significant improvements: range selection and reduced-error pruning. Range selection allows numerical (continuous-value) attributes in addition to the nominal attributes supported by ID3. Break-points are chosen to maximise the information gain, and unlike nominal attributes, numerical attributes do not lose their information value after being used higher up in the decision tree; this is due to the fact that a new split may render new information, since a new question is being asked.

Reduced-error pruning is about substituting a whole subtree with a single leaf (classification), provided that it classifies the examples well enough (e.g. 95 % correct), or skipping a test that gives less than a certain amount of information. This speeds up decision-making and reduces the risk of overtraining. Further, a subtree deep down in the tree can be elevated to a higher level, if the approximation error makes it sensible.

C4.5 uses the training data and a heuristic method to achieve estimates of the approximation errors. The parameter being estimated is the confidence limit z,

such that

$$P\left(\frac{f-q}{q(1-q)/N} > z\right) = c,$$

where N is the total number of samples, $f = E/N$ is the observed error rate, q is the true error rate, and c is the confidence. In C4.5 the default confidence is 0.25. A conservative estimate of the error rate e at a node is given by

$$e = \frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}.$$

The value of z is here the number of standard deviations corresponding to the upper confidence limit. For $c = 0.25$ the z value is 0.69.

For generating a decision tree with n instances, each with m attributes, the time complexity is $O(mn \log n) + O(n \log n)^2)$ (Witten and Frank (2005) p. 198). The first term is derived from the facts that the tree is expected to reach a depth of $\log n$, and at all levels all instances and attributes must be considered, hence the mn factor. The second term comes from subtree lifting. Replacing a subtree is an $O(n)$ operation; for every tree node an error estimate must be made, after which each node must be considered for replacement. The n subtrees may be reclassified at each level of the tree, and the time cost for reclassification of one subtree is not constant, but $O(\log n)$. Thus, the total computational cost for subtree elevation is $O(n(\log n)^2)$.

In practice, C4.5 is a very fast algorithm, but there are other machine learning algorithms that perform significantly better on real data. Nevertheless, C4.5, and its modern (non-free) successor C5.0 are considered as benchmarks both for speed and for accuracy.

4.2 SVM learning

The idea behind Support Vector Machines (SVMs) is building a separating line, plane or hyperplane, which sets two different classifications apart. First, the convex hull for the instances belonging to each classification is calculated. Then the line between the points closest to the opposite hull (shortest path from set) is drawn, and the separating plane is defined as the tangent at the median of the line. So a Support Vector Machine is strictly not a machine, but a simple and powerful algorithm.

The equation of the output from a linear SVM is

$$u = \vec{w} \cdot \vec{x} - b, \tag{4.1}$$

where $\vec{w}$ is the normal vector of the hyperplane, and $\vec{x}$ is the input vector. The separating hyperplane is at $u = 0$, and the nearest points are at $u = \pm 1$. Hence, the margin m is

$$m = \frac{1}{||w||_2}. \tag{4.2}$$

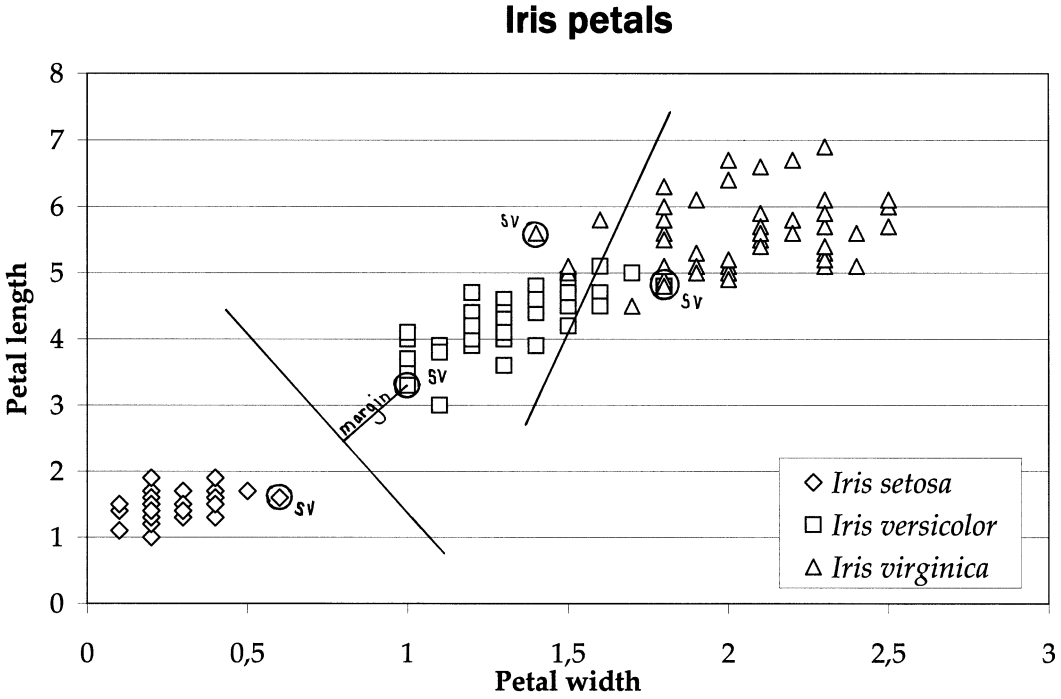

Figure 4.2: Optimal separator (diagonal lines) and support vectors (the outer points closest to the separator; marked). Petal data from Fisher (1936).

The index on the norm in the denominator above denotes that the norm to be calculated is the Frobenius norm, which in the case of a vector simply is the absolute value. Mathematically, the problem is to maximise this margin, which can be expressed through the following optimisation problem:

$$\min_{\vec{w},b} \frac{1}{2}||\vec{w}||^2 \text{ subject to } y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1, \forall i, \tag{4.3}$$

where x_i is the ith training example, and y_i is the correct output of the SVM for that example. For positive examples in a class, the value y_i is $+1$; for the negatives it is -1 (Platt, 1998).

This problem is solved using techniques described in standard optimisation textbooks, such as Lundgren et al. (2003). Platt (1998) arrives at the Quadratic Programming (QP) problem

$$\min_{\vec{\alpha}} \Psi(\vec{\alpha}) = \min_{\vec{\alpha}} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j K(\vec{x_i}, \vec{x_j}) \alpha_i \alpha_j - \sum_{i=1}^{N} \alpha_i, \tag{4.4}$$

$$o \leq \alpha_i \leq C, \forall i$$

$$\sum_{i=1}^{N} y_i \alpha_i = 0,$$

where Ψ is the objective function, solely dependent on a vector $\vec{\alpha}$ of Lagrange multipliers, K is a kernel function[1], and C is a parameter, which trades off wide margin with few margin failures.

The QP problem above (Eqn. 4.4) is the one solved by an SVM trainer. First, it was solved numerically, but Osuna et al. (1997) have proved that this problem has a simple, sequential solution. In Platt (1998), it is demonstrated that using this analytical solution saves a large – sometimes an enormous – amount of memory and computation time, and it is much less sensitive to precision-related errors than traditional methods are.

This radical method is called Sequential Minimal Optimization (SMO), and is today the standard method for training SVMs. n-ary classifications are handled by combining several separating planes to separate classes pair-wise.

The classifier trains much more slowly than C4.5, but in practical applications its accuracy vastly surpasses that of C4.5.

SMO pseudocode adapted from Platt (1998):

```
target = desired output vector
point = training point matrix

procedure takeStep(i1,i2)
  if (i1 == i2) return 0
  alph1 = Lagrange multiplier for i1
  y1 = target[i1]
  E1 = SVM output on point[i1] - y1 (check in error cache)
  s = y1*y2
  Compute L, H via equations (Pl13) and (Pl14)
  if (L == H)
    return 0
  k11 = kernel(point[i1],point[i1])
  k12 = kernel(point[i1],point[i2])
  k22 = kernel(point[i2],point[i2])
  eta = k11+k22-2*k12
  if (eta > 0)
  {
    a2 = alph2 + y2*(E1-E2)/eta
    if (a2 < L) a2 = L
    else if (a2 > H) a2 = H
  }
  else
  {
```

[1] Kernel functions express class boundaries, and are linear for linear SVMs, but can be any functions. Usually, they are polynomials or a Gaussian ('bell') functions.

```
    Lobj = objective function at a2=L
    Hobj = objective function at a2=H
    if (Lobj < Hobj-eps)
      a2 = L
    else if (Lobj > Hobj+eps)
      a2 = H
    else
      a2 = alph2
  }
  if (|a2-alph2| < eps*(a2+alph2+eps))
    return 0
  a1 = alph1+s*(alph2-a2)
  Update threshold to reflect change in Lagrange multipliers
  Update weight vector to reflect change in a1 & a2, if SVM is linear
  Update error cache using new Lagrange multipliers
  Store a1 in the alpha array
  Store a2 in the alpha array
  return 1
endprocedure

procedure examineExample(i2)
  y2 = target[i2]
  alph2 = Lagrange multiplier for i2
  E2 = SVM output on point[i2] - y2 (check in error cache)
  r2 = E2*y2
  if ((r2 < -tol && alph2 < C) || (r2 > tol && alph2 > 0))
  {
    if (number of non-zero & non-C alpha > 1)
    {
      i1 = result of second choice heuristic (section 2.2)
      if takeStep(i1,i2)
        return 1
    }
    loop over all non-zero and non-C alpha, starting at a random point
    {
      i1 = identity of current alpha
      if takeStep(i1,i2)
        return 1
    }
    loop over all possible i1, starting at a random point
    {
      i1 = loop variable
      if (takeStep(i1,i2)
        return 1
    }
  }
```

```
  return 0
endprocedure

main routine:
  numChanged = 0;
  examineAll = 1;
  while (numChanged > 0 | examineAll)
  {
    numChanged = 0;
    if (examineAll)
      loop I over all training examples
        numChanged += examineExample(I)
    else
      loop I over examples where alpha is not 0 & not C
        numChanged += examineExample(I)
    if (examineAll == 1)
      examineAll = 0
    else if (numChanged == 0)
      examineAll = 1
  }
```

$$L = \max(0, \alpha_2 - \alpha_1), \quad H = \min(C, C + \alpha_2 - \alpha_1) \quad \text{(Pl13)}$$
$$L = \max(0, \alpha_2 - \alpha_1 - C), \quad H = \min(C, \alpha_2 + \alpha_1) \quad \text{(Pl14)}$$

4.3 Logistic Model Trees

Like C4.5, this machine learning method builds decision trees, but unlike C4.5 several variables may be used at a time. It builds on logistic regression, which in essential is a generalised linear model for binary dependent variables. Its link function is the logit function, and the errors are binomially distributed.

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \alpha + \beta_1 x_{1,i} + \ldots + \beta_k x_{k,i};\ i = 1, \ldots, n, \tag{4.5}$$

where

$$p = P(Y_i = 1).$$

Solving (4.5) for p, the equation

$$p = P(Y_i = 1|X) = \frac{e^{\alpha+\beta_1 x_{1,i}+\ldots+\beta_k x_{k,i}}}{1 + e^{\alpha+\beta_1 x_{1,i}+\ldots+\beta_k x_{k,i}}} \tag{4.6}$$

is obtained. The α and β parameters are usually estimated using the Maximum Likelihood method. See an inference theory textbook, such as Blom and Holmquist (1998) for a description of the ML method.

The logistic regression model is quite stable, but it has the disadvantage of being restricted to linear patterns in the data. Of course, a non-linear pattern can be translated into a linear one via variable substitution, but searching polynomial spaces, which would be a very time-consuming operation[2], is not necessary.

Instead, a tree model with linear logistic regression models as branching functions can be used. An ordinary tree model has the disadvantage of being prone to overtraining, if the example dataset is large. For little or noisy example data, the LMT algorithm uses a simple linear regression model, whereas it for larger datasets uses a tree model.

The LogitBoost step performs additive logistic regression. At each iteration, a simple logistic regression function is fit, using all the attributes, while minimising the total errors, after which the function is added to the model. If the algorithm is run until it converges, the result will be an ML[3] multilogistic regression model. But since the model is to be used to classify unseen data, letting the boosting step converge may overfit the model, decreasing classifier performance. Therefore, cross-validation is used to decide when the boosting shall stop.

When no more structures in the data can be modeled using a linear logistic regression function, the boosting process terminates. However, there may still be structures of the data, which can be described by linear logistic models (LLMs), if only a part of the data is considered. Such subsets can be found using standard decision tree criteria, e.g. information gain.

Once no improvement can be made by considering only a part of the parameter space, the algorithm starts splitting the data, boosting each subset individually and separately refining the model for them. Even in this process, cross-validation is used to decide the appropriate amount of refinement.

The splitting process is applied recursively until the subsets become too small. It is certain that the model will be overfit; therefore pruning the model tree is essential. Once again, cross-validation is the appropriate technique to maximise classifier performance on arbitrary data. The final result is a small but accurate tree with LLMs at the leaves; sometimes[4] it is a simple LLM or a single-node combination of LLMs.

[2] Constructing all polynomials of degree m from an n-dimensional space yields a linear space of dimension $\binom{n}{m}$.

[3] Maximum Likelihood

[4] See above w.r.t. noisy or small example sets.

LMT pseudocode from Landwehr et al. (2003)

```
LMT(examples){
    root = new Node()
    alpha = getCARTAlpha(examples)
    root.buildTree(examples, null)
    root.CARTprune(alpha)
}

buildTree(examples, initialLinearModels) {
    numIterations = crossValidateIterations(examples, initialLinearModels)
    initLogitBoost(initialLinearModels)
    linearModels = copyOf(initialLinearModels)
    for i = 1...numIterations
        logitBoostIteration(linearModels,examples)
    split = findSplit(examples)
    localExamples = split.splitExamples(examples)
    sons = new Nodes[split.numSubsets()]
    for s = 1...sons.length
        sons.buildTree(localExamples[s], nodeModels)
}

crossValidateIterations(examples,initialLinearModels) {
    for fold = 1...5
        initLogitBoost(initialLinearModels)
        //split into training/test set
        train = trainCV(fold)
        test = testCV(fold)
        linearModels = copyOf(initialLinearModels)
        for i = 1...200
            logitBoostIteration(linearModels,train)
            logErrors[i] += error(test)
    numIterations = findBestIteration(logErrors)
    return numIterations
}
```

4.4 Naive Bayes

The Naive Bayes method generates rules based on Bayes's rule of conditional probability, Equation 4.7, where C_i denotes the classes, and A denotes a set of conditions on the attributes. All attributes are used to make the decisions, and they are considered to be independent from each other and of equal importance.

$$P(C_i|A) = \frac{P(C_i) \cdot P(A|C_i)}{\sum_{j=1}^{n} P(C_j) \cdot P(A|C_j)} \tag{4.7}$$

In practice, only the numerator of this equation is interesting; the denominator does not depend on the choice of i, and can therefore be regarded as a constant. Since the attributes are considered to be independent, the probability that an observation is of class C_i, given the attribute values A_j, is

$$P(C_i|A) = \frac{1}{Z} \cdot P(C_i) \prod_{j=1}^{n} P(A_j|C_i).$$

The parameters needed in the statistical model are estimated from the training data using e.g. the maximum likelihood method or Bayesian inference.

To determine the class in which an observation belongs, one can use the equation

$$\ln \frac{P(C_i|A)}{P(\neg C_i|A)} = \ln \frac{P(C_i)}{P(\neg C_i)} + \sum_{j=1}^{n} \ln \frac{P(A_j|C_i)}{P(A_j|\neg C_i)},$$

which statisticians often do, and they call it the *log-likelihood ratio*. The log-likelihood ratio for a given observation A and class C_i is positive if and only if A belongs to C_i (Wikipedia, 2006).

4.5 Evaluating Automatic Classification Algorithms

Three common measures of classifier performance are accuracy, precision and recall. They are defined as:

$$Accuracy = \frac{Number\ of\ correctly\ classified\ samples}{Total\ number\ of\ samples}$$

$$Precision(C) = \frac{Number\ of\ samples\ correctly\ classified\ as\ class\ C}{Total\ number\ of\ samples\ classified\ as\ class\ C}$$

$$Recall(C) = \frac{Number\ of\ samples\ classified\ as\ class\ C}{Total\ number\ of\ samples\ in\ class\ C}$$

Accuracy is an overall measure of classifier performance, whereas *precision* deals with the classifier's tendency to make mistakes by over-generalising, and *recall* is the ability to correctly classify all instances belongnig to a certain class.

Chapter 5

The Processing Pipeline

5.1 Tools employed

For the study of machine learning techniques for the Direkt Profil system the following tools have been employed:

- Java (with custom specialised tools)
- Weka
- Direkt Profil

5.2 Making profiles

From the large CEFLE corpus, in XML format, the 'Italie' subcorpus was extracted in a text editor and read with an in-house tool, XMLCorpusReader. The output format of the XMLCorpusReader is LearnerText objects, basically an objectified version of the XML dataset, which can then be analysed using external, custom profilers or Direkt Profil.

Two examples of custom profilers are TTR (Type-Token Ratio) and Bag of Words. TTR, which calculates the ratio between unique words in the text and the total number of words, a ratio which is believed to be lower for more advanced learners than for beginners.

$$TTR = \frac{Number\ of\ unique\ words}{Total\ number\ of\ words}$$

Another one is Bag of Words, which collects every unique word of every text in the Italie subcorpus (or any XML corpus file) and for each text counts how many times every word in the corpus occurs. For each text, the Bag of Words profiler

indicates which words that occur in it. Significant usage of infrequent words may imply a higher learner stage, while misspelling common words, or using Swedish or English words due to a lack of vocabulary, is more likely on the lower stages. This can be observed in the following beginner text, where all the bracketed words are Swedish:

> Deux filles, Marie et Louise. Marie a grosse fille, blond hereuex et Louise a petite fillem marron hereuex. Ils (vill) travaillie en Italie. Ils (vill) encontre deux garcons et (ligga på) playa. Aujourd hui ils (packar) son bagage et travallie en Italie. Ils arrivee a hotel et (går upp till) son chambre. Beau vy! Ils (går ner till) playa et soleil. (Senare) ils (äter) dejuner, sphagetti boulognaise. Ils (sätter sig) dans la bar. Ils (ser) deux garcons ... amore! Ils parle et danse (hela) nuit. Deux garcons, Carlos et Diego. (Nästa dag) (åker)Marie, Louise, Carlos et Diego un sightseeing et shopping dans le market. La (kvällen) (sitter) Marie et Carlos et vu le soleil (nedgång). Louise ET Diego (äter) dejeuner et (dricker) wine. Ils (lyssnar också) dans la musique. Romantique! (Nästa dag)

However, the real crown jewel is Direkt Profil, a servlet based system that takes text via a web interface and presents counter values for several dozens of grammatical features. An example Direkt Profil session is shown in Figure 3.2, with a learner text and some grammatical features annotated. More details about the Direkt Profil annotation are described in Kostadinov (2005).

5.3 Auto-assessing profiles

There are two ways for a computer to classify a piece of text, given statistical data about it, one being with pre-defined (hard-coded) rules; the other one being machine-learning, where the computer finds out the rules via statistical methods.

Hard-coded rules have the advantage that they are easy to implement, since the programmer just needs to write a classification function, based on e.g. decision trees or mathematical formulae: a rather straightforward procedure. The main disadvantage is that constructing reasonable decision trees and formulae require thorough studies in the subject matter, in this case Swedish high-school learners' acquisition of written French. A comprehensive corpus is needed, preferably tracking the development of a large number of learners over a long time, and even more importantly, the learner texts must be manually analysed and classified.

Using the machine learning methods described in Chapter 4, all available in Weka, I tested different mathematical models, to see which one best predicts the learner's stage. Letting the computer do the hard work is both quick and flexible, especially for freely written texts with little or no guidance. Yet, the human ability to make guesses and set up hypotheses remains an invaluable asset in the search for the

ultimate profiler, if one exists. Even if there is one, it will not always agree with human examiners.

Since the analysed corpus is small, cross-validation is useful in evaluating classifier performance without losing any useful information. In the form of cross-validation employed in my experiments, 10-fold cross-validation, the entire corpus is divided into ten subcorpora, of which nine are used as the training data, and the tenth is used as the testing set. The ten subcorpora take turns being the testing set, so the entire corpus is used both for training and testing, yielding ten performance scores, which are averaged to form a single overall performance score.

5.4 Introduction to my homemade tools

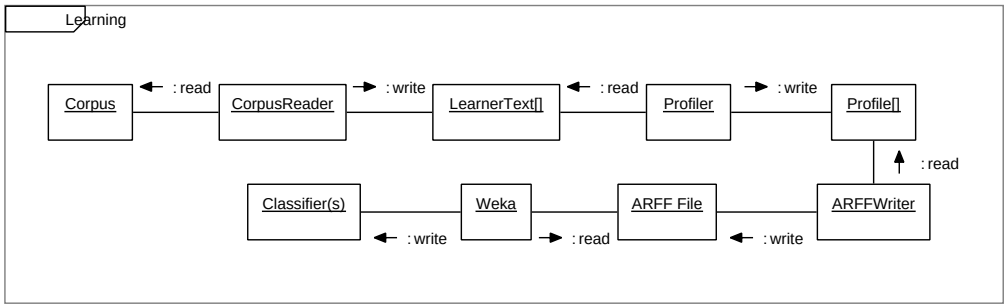

Figure 5.1: Outline of the learning process

5.4.1 The learning process

When the system learns how to classify a text, it reads the corpus into an array of learner texts, from which it then extracts one feature vector for each text. It also takes notice of the manual pre-classification, which it saves in a special field of a Profile object, in which the learner's name and the feature vector for the learner's text also are stored.

These data are re-written into an ARFF file (see Section B.1), which can be read by Weka, the program used to fit classifying models. The entire process is illustrated in Figure 5.1.

5.4.2 The application process

If there exists a well-fit classifier, any learner text similar to those used to train the system, can be automatically classified in this process. Just like in the learning process, the text must be translated into a feature vector before can be processed

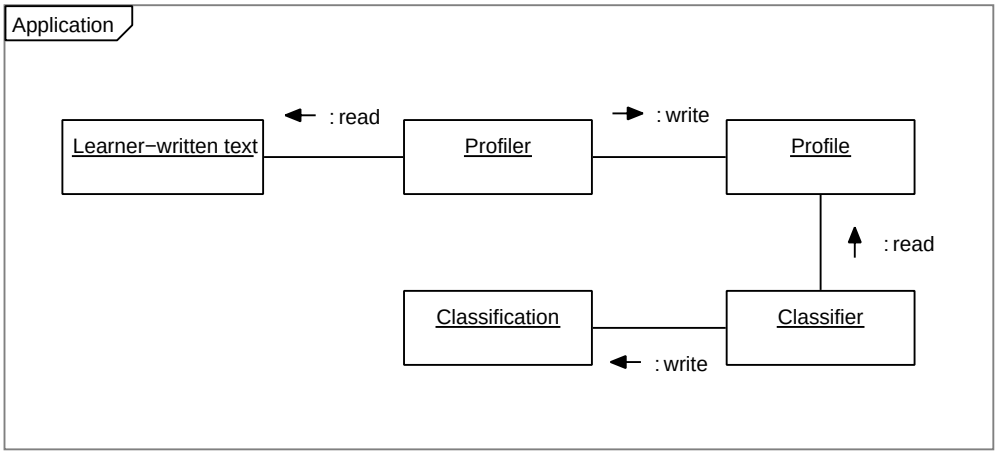

Figure 5.2: Outline of how classifiers could be integrated with the DP system

any further. Naturally, there is no point in applying a classifier to a parameter space for which it has not been trained, so the profile must be post-processed in exactly the same way as the training data. Then the classifier can classify the unknown text, assigning a proficiency classification to it. This process is described in Figure 5.2.

5.4.3 Readers

XMLCorpusReader

This module reads an entire XML corpus, and divides it into LearnerText objects, over which profilers can iterate. In the future, this module could be used to send individual texts from a corpus to Direkt Profil.

5.4.4 Profilers

Profilers are a kind of objects, which contain a profiling function, taking as their input a student name and the text that said student wrote. The output consists of a Profile object, which contains the profile type, e.g. *Lexical*; the student's name, and a list of variables (features).

SimpleProfiler (TTR)

This profiler is based on the type-token ratio feature, which is the ratio of unique words to the total number of words in the text. It contains these features:

1. Total number of words
2. Number of unique words
3. Type-token ratio (TTR)

BagOfWords

This profiler uses the bag of words technique, yielding a very large, and often sparse, search space. As texts are added to the working space of this profiler, words not having previously occurred are added to the profiler's universal dictionary, which is a simple sorted set, and the counts of each word occurring in each text are saved in a BagOfWords data structure with the student's name and a word counts hash table.

So internally, there is one large dictionary, shared between all bags of words, and each bag of words has its own dictionary with word counts. The reason for this design decision is to conserve memory, since no student uses even a significant part of all the words[1] used in all the texts.

LexicalProfiler

This profiler looks up every word written in the text in the Lexique3 database[2], and counts the total number of occurrences of words in these categories:

- Words among the 1,000 most common ones
- Words 1,001–3,000 in order of frequency
- Less frequent words (3,001–)
- Words that do not exist in the dictionary

The parsing features of Direkt Profil are not being used, thus it is not possible to tell ambiguous word forms apart, but they are instead treated as the same words.

[1] including the occasionally seen misspellings and Swedish or English substitutions

[2] Special two-column word frequency lists, originating from books and films, with either individual counts for each form or combined counts for the same word regardless of its form, have been generated from the original ASCII multi-column version with a Perl script.

XMLCounterProfiler

This profiler assembles the counters given by the Direkt Profil system into a Profile object. Using it requires a manual copy-paste-analyse-save procedure with both the corpus and the Direkt Profil web interface open. The exported counter files (profile source data) are best placed in one folder, on which either the `XMLCounters2ARFF` or `CombinedProfilers` tools can be used.

This is the DTD of an XML counter file:

```
<!DOCTYPE counters [
   <!ELEMENT counters (counter+)>
   <!ELEMENT counter EMPTY>
   <!ATTLIST counter
       id        CDATA    #REQUIRED
       descr     CDATA    #REQUIRED
       value     CDATA    #REQUIRED>
]>
```

5.4.5 Writers

ARFFWriter

This writer writes an ARFF file (see Section B.1), given a set of Profiles, which are added one by one. Profile type is transferred; student names are for practical reasons[3] not.

Profile.writeProfiles()

The Profile class contains a statical method for writing a set of profiles to an XML file and one to read them back. Since this functionality has not been used for this project, but is rather a preparation for the future stages, it is rather unstable.

5.5 Experiments

The earliest experiments were comparing classifier performance using data from the TTR profiler only with the C4.5 decision tree and SMO support vector machine algorithms. Further experiments were conducted with Bag of Words and lexical profilers, and finally TTR, lexical and morphosyntactical were combined, and two more machine learning schemes, LMT and Naive Bayes, were tested, in addition to the previous C4.5 and SVM schemes.

[3] Introducing an irrelevant nominal attribute would confuse the classifier.

Figure 5.3: The Combined Profilers experiment GUI

Generally, the tools which create the ARFF files used for the experiments are command-line programs. However, to facilitate the Combined Profilers experiment a GUI, *ProfilersGUI*, has been created. It lets the user graphically choose the corpus file, counter files location, ARFF and XML output files, and dictionary to be used. After the ARFF file has been prepared, it can be located and processed using the Weka Explorer.

Chapter 6

Quantitative results

Some definitions repeated

$$Precision(C) = \frac{Number\ of\ samples\ correctly\ classified\ as\ class\ C}{Total\ number\ of\ samples\ classified\ as\ class\ C}$$

$$Recall(C) = \frac{Number\ of\ samples\ classified\ as\ class\ C}{Total\ number\ of\ samples\ in\ class\ C}$$

$$TTR = \frac{Number\ of unique\ words}{Total\ number\ of\ words}$$

6.1 TTR

This profiler uses three features:

1. Total number of words
2. Number of unique words
3. Type-token ratio (TTR)

The following classifier performance data was obtained when using the TTR profiler:

6.1.1 C4.5

```
Correctly Classified Instances          53               50.4762 %
Incorrectly Classified Instances        52               49.5238 %
```

```
  a  b  c  d  e   <-- classified as
  0  6  4  0  0 |  a = 1
  6 14  9  0  0 |  b = 2
  2  7 14  3  2 |  c = 3
  0  0  6  6  4 |  d = 4
  0  0  1  2 19 |  e = 6
```

Here only three features (total number of words, number of unique words, and the TTR) are taken into consideration, and a simple classifier is used. Yet, the system get more than half of the instances right. It is especially strong on stage 6, with a precision of 76 % and an 83 % recall, and the weakest stage is 1, with zero precision and recall. On the other stages, precision and recall values are in the 50 % range and below.

6.1.2 SVM

```
Correctly Classified Instances          56               53.3333 %
Incorrectly Classified Instances        49               46.6667 %

  a  b  c  d  e   <-- classified as
  0 10  0  0  0 |  a = 1
  0 23  6  0  0 |  b = 2
  0 11 15  0  2 |  c = 3
  0  0 12  0  4 |  d = 4
  0  1  3  0 18 |  e = 6
```

With an SVM, better general performance is achieved. Grave misclassifications are rarer, but stage 1 still has zero precision and recall. Now, this has also happened to stage 4, probably due to an overlap, as is the case in the right separating line of Figure 4.2.

6.2 Bag of Words

The Bag of Words profiler counts how many times each word in the corpus occurs in each text. Many counts are zero, since not every word in every spelling variation, correct or incorrect, occurs in any text.

In this section, data for stage 6 are missing, since these experiments were conducted in an early stage of the thesis project, and due to low performance, it has not been further developed, unlike the rest of the classification test bed; therefore, it is no more compatible with it, and fixing it afterwards would take a prohibitive amount of time.

6.2.1 C4.5

```
Correctly Classified Instances          43               50      %
Incorrectly Classified Instances        43               50      %

  a  b  c  d   <-- classified as
  4  6  1  0 |  a = 1
  4 18  6  1 |  b = 2
  3 12 11  3 |  c = 3
  1  2  4 10 |  d = 4
```

These results are similar to those achieved in the corresponding TTR experiment. However, this classifier, unlike the TTR classifiers, also finds texts on stage 1, be it with rather low performance. Stage 4 performance is also improved when bags of words are used.

6.2.2 SVM

```
Correctly Classified Instances          50               58.1395 %
Incorrectly Classified Instances        36               41.8605 %

  a  b  c  d   <-- classified as
  0 11  0  0 |  a = 1
  0 24  5  0 |  b = 2
  0 13 15  1 |  c = 3
  0  1  5 11 |  d = 4
```

Once again, no texts are classified as stage 1, but precision on stage 4 is very good, as is recall on stage 2.

6.3 LexicalProfiler

The Lexical profiler uses these features:

- Words among the 1,000 most common ones
- Words 1,001–3,000 in order of frequency
- Less frequent words (3,001–)
- Words that do not exist in the dictionary

Here are the results achieved by two types of classifiers using the LexicalProfiler data:

6.3.1 C4.5

```
Correctly Classified Instances          33               30.5556 %
Incorrectly Classified Instances        75               69.4444 %

  a  b  c  d  e   <-- classified as
  1  5  3  1  1 |  a = 1
  5 11  7  4  2 |  b = 2
  1  8  6  5  9 |  c = 3
  0  2  4  6  5 |  d = 4
  1  2  6  4  9 |  e = 6
```

Here, it seems like C4.5 and the lexical profiler simply do not work well together. Performance is worse than when the simple TTR and word counts profiler was used.

6.3.2 SVM

```
Correctly Classified Instances          37               34.2593 %
Incorrectly Classified Instances        71               65.7407 %

  a  b  c  d  e   <-- classified as
  0  7  4  0  0 |  a = 1
  0 18 10  0  1 |  b = 2
  0  9 14  0  6 |  c = 3
  0  2 15  0  0 |  d = 4
  0  3 14  0  5 |  e = 6
```

These figures are not impressive either, and the well-known zero precision and recall figures for stages 1 and 4 in the SimpleProfiler experiment reoccur. Still, general performance is marginally better than with C4.5, but it is not good enough to be useful.

6.4 Direkt Profil 2.0 percentages

For this profiling method, only the counter values from Direkt Profil 2.0 have been used[1]. The percentages were created by post-processing the raw counter values in the Weka Explorer[2].

[1] The complete list of counters can pe found on page 13*ff*.
[2] See Section B.2 on page 60*ff*.

6.4.1 C4.5

```
Correctly Classified Instances          51               49.0385 %
Incorrectly Classified Instances        53               50.9615 %

  a  b  c  d  e   <-- classified as
  5  3  1  0  0 |  a = 1
  2 13 10  4  0 |  b = 2
  0  9 13  6  0 |  c = 3
  0  0  8  4  4 |  d = 4
  0  1  3  2 16 |  e = 6
```

On the contrary, DP 2.0 and C4.5 bring the performance figures known from the first two experiments. While it is somewhat disappointing that having 20 times more attributes does not give near-perfect results, there could be relationships that C4.5 does not detect.

6.4.2 SVM

```
Correctly Classified Instances          67               64.4231 %
Incorrectly Classified Instances        37               35.5769 %

  a  b  c  d  e   <-- classified as
  5  4  0  0  0 |  a = 1
  1 19  9  0  0 |  b = 2
  0  6 17  3  2 |  c = 3
  0  1  5  9  1 |  d = 4
  0  0  2  3 17 |  e = 6
```

This proved to be true; DP 2.0 and SVM yield the best performance achieved until now. Only a few texts are classified far off from where they should be, and precision goes from 52 (stage 3) to 88 % (stage 1), while recall goes from 56 (stage 1) to 77 % (stage 6).

6.5 Combined profilers

Combining the values obtained from the TTR profiler, LexicalProfiler and Direkt Profil gave the following results:

6.5.1 C4.5

```
Correctly Classified Instances          58               55.7692 %
Incorrectly Classified Instances        46               44.2308 %
```

```
  a  b  c  d  e   <-- classified as
  5  3  1  0  0 |  a = 1
  1 17  7  3  1 |  b = 2
  1  6 14  7  0 |  c = 3
  0  0  6  6  4 |  d = 4
  0  1  3  2 16 |  e = 6
```

When TTR and lexical attributes are added to the Direkt Profil 2.0 attribute space, system performance improves somewhat. On stages 2, 3, and 4, 4, 1 and 2 more texts, respectively, have been correctly classified.

6.5.2 SVM

```
Correctly Classified Instances          66               63.4615 %
Incorrectly Classified Instances        38               36.5385 %

  a  b  c  d  e   <-- classified as
  6  3  0  0  0 |  a = 1
  1 19  9  0  0 |  b = 2
  0  6 17  4  1 |  c = 3
  0  1  5  9  1 |  d = 4
  0  1  2  4 15 |  e = 6
```

Surprisingly, adding the five extra attributes from Simple- and LexicalProfiler to those from DP2.0 impairs performance somewhat: two native-speaker texts are re-classified at lower stages; one of them at an absurd stage 2, whereas one more stage 1 text is correctly classified.

6.5.3 LMT

```
Correctly Classified Instances          63               60.5769 %
Incorrectly Classified Instances        41               39.4231 %

  a  b  c  d  e   <-- classified as
  4  5  0  0  0 |  a = 1
  2 18  9  0  0 |  b = 2
  0 10 13  5  0 |  c = 3
  0  1  3  9  3 |  d = 4
  0  1  0  2 19 |  e = 6
```

Generally, this simple statistical model[3] performs closely comparably to the SVM model. Stage 2 precision and stage 6 recall have significantly improved, but three correct classifications have been lost.

[3] The result from Weka was a multilogistic regression model without any tree nodes.

6.5.4 Naive Bayes

```
Correctly Classified Instances          68               65.3846 %
Incorrectly Classified Instances        36               34.6154 %

  a  b  c  d  e   <-- classified as
  5  3  1  0  0 |  a = 1
  1 22  6  0  0 |  b = 2
  0  9 14  3  2 |  c = 3
  0  0  6  9  1 |  d = 4
  0  0  1  3 18 |  e = 6
```

Here is the real surprise: The simple and fast statistical naive Bayes model performs best amongst those tested! Even if it in practice is much faster than LMT and SVM, it performs on the same level as they do; precision and recall figures change for individual classes, but for all examples in the corpus, this classifier made the highest number of correct classifications.

Chapter 7

Evaluation

These results are within the range of the expected, given the difficulty of the classification problem. Even human experts would disagree about some of the classifications in the corpus. But that is not the only reason for the less-than-perfect precision and recall values.

Another obstacle in the machine learning and automatic classification process is the large number of error sources: First of all, it can be discussed whether the learner stage hypothesis is correct. Are there really four distinct developmental stages observable in the data which Direkt Profil and my tools give? Further, the texts have been manually classified and annotated for testing the parser and classifiers, and the Direkt Profil system is not bug-free, and even if it were, there would still be the problem of ambiguities not resoluble through syntactical analysis, but only if semantics are taken into account.

7.1 Summary

In this thesis, the concepts of machine learning have been discussed, and the Direkt Profil grammatical profiling system, in addition to a simple profiling framework written by me, has been introduced. Four machine learning schemes: C4.5, support vector machines, logistic model trees and naive Bayes, have been tested, and practically SVMs have given the best and most consistent results. Precision and recall figures are approaching 90 % for native speakers, when the TTR measure, lexical profiling and the output from Direkt Profil are combined, and the figures do not go below 50 % for any learner stage. Once Direkt Profil 2.0 could be used, classification accuracy improved somewhat.

7.2 Suggestions for the future

Certainly, more research is needed, as well as further refinement of the Direkt Profil system. Some categories and attributes from the theories in Bartning and Schlyter (2004) are still missing, and there the system makes many mistakes in some categories (far from perfect precision/recall).

Appendix A

The LREC 2006 article

after pp. 565–570 of the 2006 *Proceedings of the 5th International Conference on Language Resources and Evaluation*, of which I wrote Section A.5.

CEFLE and Direkt Profil: a New Computer Learner Corpus in French L2 and a System for Grammatical Profiling

Jonas Granfeldt*, Pierre Nugues†, Malin Ågren*, Jonas Thulin†, Emil Persson*, Suzanne Schlyter*

*Centre for languages and literature – Lund university
Box 201, S-221 00 Lund, Sweden
{Jonas.Granfeldt, Malin.Agren, Suzanne.Schlyter}@rom.lu.se,
Emil.Persson@telia.com
† Department of Computer science – Lund Institute of Technology
Box 118, S-221 00 Lund, Sweden
Pierre.Nugues@cs.lth.se, f00jt@efd.lth.se

Abstract

The importance of computer learner corpora for research in both second language acquisition and foreign language teaching is rapidly increasing. Computer learner corpora can provide us with data to describe the learner's interlanguage system at different points of its development and they can be used to create pedagogical tools. In this paper, we first present a new computer learner corpora in French. We then describe an analyzer called *Direkt Profil*, that we have developed using this corpus. The system carries out a sentence analysis based on developmental sequences, i.e. local morphosyntactic phenomena linked to a development in the acquisition of French as a foreign language. We present a brief introduction to

developmental sequences and some examples in French. In the final section, we introduce and evaluate a method to optimize the definition and detection of learner profiles using machine-learning techniques.

A.1 Introduction

The importance of computer learner corpora (CLC) for research in both second language acquisition and foreign language teaching is rapidly increasing. As pointed out by Granger (2004), CLCs can serve different purposes in the research process. They can provide us with data to describe the learner's interlanguage system at different points of its development and they can be used to create pedagogical tools. CLCs might also be used indirectly to improve classroom practice.

In this paper, we first present a new CLC in French, the CEFLE corpus. We then describe an analyzer called *Direkt Profil*, that we have developed using this corpus. The system carries out a sentence analysis based on developmental sequences, i.e. local morphosyntactic phenomena linked to a development in the acquisition of French as a foreign language. The objective of the program is to establish a learner profile based on the grammatical features of the input text. We present a brief introduction to developmental sequences and some examples in French. We also present and evaluate some recent developments in Direkt Profil. In the final section, we introduce and evaluate a method to optimize the definition and detection of learner profiles using machine-learning techniques.

A.2 The CEFLE Corpus

The Lund CEFLE Corpus (*Corpus Écrit de Français Langue Étrangère*) is a written corpus of texts in French as a foreign language. This longitudinal corpus contains approximately 400 texts (100,000 words) written by Swedish learners of French with different levels of proficiency and by French native speakers in a control group. CEFLE is the result of a study that surveyed 85 learners of French in the Swedish high school throughout the academic year 2003/2004. During this period, each learner wrote four texts in French at two months intervals. The aim of this study was to analyze the morphosyntactic development in written production. The control group of 22 native speaking adolescents is completing this material.

The foreign language learners in the CEFLE corpus have Swedish as their mother tongue and they are advanced L2 learners of English. French corresponds to their second or third foreign language. They all learn French in a traditional instructional setting at the Swedish high school. The beginner learners are attending their first year of French when writing the first text. The most advanced learners started their fifth year of French at the beginning of the study.

CEFLE contains texts from four different tasks, which were created to elicit written data as spontaneously as possible from all kinds of learners. Two different task

CEFLE corpus			Subcorpus *Le voyage en Italie* (averages)			
Task name	Elicitation type	Words		TL	SL	Vocd
Homme	Pictures	17,260	Stage 1 (N=10)	127	7.0	40.5
Souvenir	Pers. Narrative	14,365	Stage 2 (N=29)	175	8.4	53.5
Italie	Pics	30,840	Stage 3 (N=39)	276	9.9	60
Moi	Pers. Narrative	30,355	Stage 4 (N=17)	369	11.8	74
Total		92,820	Control (N=22)	334	9.7	104

Table A.1: General description of the CEFLE corpus and the subcorpus *Le voyage en Italie* (TL = Text length; SL = Sentence length; Vocd = vocabulary diversity).

types were used: (1) story telling tasks based on picture sequences, (2) descriptive narratives based on personal experiences. The texts *L'homme sur l'île* 'The man on the island' and *Le voyage en Italie* 'The trip to Italy' are representing the first task type, while *Moi, ma famille et mes amis* 'Me, my family and my friends' and *Un souvenir de voyage* 'Memory of a journey' are representing the personal narratives. All texts were written on a computer using plain text formatting.

The texts from one of the four elicitation procedures, *Le voyage en Italie* 'The journey to Italy', has been used as a subcorpus receiving special attention in several respects: a cross-sectional linguistic analysis was carried out on this material (Ågren, 2005) and these texts are used in the work with Direkt Profil. Developmental sequences based on morphosyntactic criteria (Bartning and Schlyter, 2004) were used to place the learner texts on four levels of development: stage 1 (initial), stage 2 (post-initial), stage 3 (intermediate), and stage 4 (pre-advanced). This part of the corpus is annotated for a specific set of lexical or syntactic phenomena using the XML format. A brief description of the linguistic levels in the subcorpus is presented in Table A.1. Vocd is a measure of vocabulary diversity developed on the basis of the traditional type-token ratio by Malvern et al. (2004). A high Vocd value is interpreted as a rich vocabulary.

A.3 Developmental Sequences in French L2

Developmental sequences are features and constructions linked to a development over time in the grammar of the language learner. Beginning in the 1970s, the so-called *morpheme order studies* identified the order in which grammatical morphemes like *–ing*, *the*, *a* and *'s* simply appeared in English spoken by nonnative speakers (Dulay and Burt, 1974; Bailey et al., 1974). Some argued that these sequences were universal referring to them as the "natural order of development". More recently Pienemann (1998) framed the development of German L2 in six grammatically defined stages. The underlying rationale behind these kinds of proposals is the idea of the learner language as its own grammatical system, an *interlanguage* (Selinker, 1972), independent from the target language system. Theories differ however, when it comes to accounting for the observed development.

According to the *Processability Theory* (Pienemann, 1998), currently the most detailed account for sequences of L2 development, the learner can only produce structures that can be processed. In this theory, acquiring to produce linguistic structures is seen as a process of automatization where each step in the development builds on the previous one. Development is constrained by limits of the working memory. Automatizing the processing will free working memory capacity that will in turn enable the learner to process and produce increasingly more complex structures. Consider an example from our corpus: *Un soir les filles mange dans un restaurant*. In French, a language with rich subject-verb agreement in the written system, the writer must store in memory the features of the subject when producing the verb. In this case, the subject is in 3rd person plural (*les filles*) and the verb should be marked accordingly (*mangent*). In the example above, the learner has instead used a default form, the 3rd person singular (*mange*) thus producing a typical learner error. What is important for us, is that the theory of developmental sequences and developmental stages in the learning of foreign language predicts that learners in their production of S-V agreement and other features will show optionality. For the relevant features, optionality is developmental in nature and a characteristic of transitional stages. Therefore, we expect a lack of S-V agreement to be more important in initial stages of development, and perhaps with more variation, and we expect it to disappear in the advanced stage.

Stages	1	2	3	4	5	6
% of finite forms of lexical verbs in obligatory contexts	50-75	70-80	80-90	90-98	100	100
% of 1st person plural S-V agreement (*nous V-ons*)	–	70-80	80-95	100	100	100
% 3rd pers plural agreement with irregular lexical verbs like *viennent, veulent, prennent*	–	–	a few cases	≈ 50	few errors	100
Object pronouns (placement)	–	SVO	S(v)oV	SovV app.	SovV prod	acquired (also *y* and *en*)
% of grammatical gender agreement	55-75	60-80	65-85	70-90	75-95	90-100

Table A.2: Developmental sequences from Bartning and Schlyter (2004). Legend: – = no occurrences; app = appears; prod = productive advanced stage.

Developmental sequences have been studied in spoken L2 French by Bartning and Schlyter (2004). They defined about 25 different morphosyntactic features and proposed a definition of their development over time in adult Swedish learners of French. Taken together, these features shape six grammatical profiles – ranging from beginner to very advanced learners. Examples of features are given in Table A.2. As the language learner moves towards an increasing automatization of the target language, the produced structures become increasingly more complex

and more targetlike. Developmental sequences describe in linguistic terms this process.

A.4 Direkt Profil

Direkt Profil (DP) is the system we are developing to identify, annotate, quantify, and display the specific linguistic constructions connected to a development over time in foreign language French. In other words, DP analyzes the learners' texts for structures occurring in developmental sequences (see Table A.2). The CEFLE corpus (see above Section A.2) serves as a development and test corpus in the implementation of DP. The overall architecture of Direkt Profil was described in a previous paper (Granfeldt et al., 2005b) and we will limit our presentation here to some recent developments.

Verb groups and noun groups represent the essential grammatical support of our annotation. The majority of syntactic annotation standards for French take such groups into account in one way or another. The PEAS annotation scheme (Gendner et al., 2004) is a consensual example that reconciles a great number of annotations. However, in their present shape, these standards are insufficient to mark up constructions of Table A.2, many of which are specific to foreign language writers.

On the basis of the linguistic constructions in Bartning and Schlyter (2004), we developed our own annotation scheme. The current version of DP, 1.5.4, detects four types of syntactic groups, nonrecursive noun groups, verb groups, prepositional groups, and conjunctions, that it annotates using the XML format.

The DP architecture is a cascade of five layers. The first layer corresponds to tokenization of the text. The second layer annotates prefabricated expressions or sentences (e.g. *je m'appelle* 'my name is'). These structures correspond to linguistic expressions learned "by heart" in a holistic manner. It has been shown that they have a great importance in the first years of learning French.

The third layer corresponds to a chunk annotation of the text, restricted to the phenomena to identify. This layer marks up the verb and noun groups. As in PEAS, the verb group incorporates the subject clitic pronouns. The XML element `segment` marks the groups. Table A.3 presents an evaluation in precision and recall of the chunking layer.

The fourth layer uses a `tag` element with attributes to indicate the lemma, the part of speech, and the grammatical features. For the verb group, the sentence *Ils parlons dans la bar* extracted from the learner text above receives the following annotation:

```
<segment class="c5131"><tag pos="pro:  nom:pl:p3:mas">Ils</tag>
<tag pos="ver:  impre:pl:p1">parlons</tag></segment> dans la bar.
```

The `c5131` class is interpreted as "finite lexical verb no agreement".

The fifth layer counts structures typical of an acquisition stage. It uses the `counter` XML element,

```
<counter id="c5200" counter_name= "passe_compose"
rule_id="participe_4b" value="1"/>.
```

The analyzer uses manually written rules and a lexicon of inflected terms. The recognition of the group boundaries is done by a set of closed-class words and the heuristics inside the rules. It thus follows an old but robust strategy used in particular by Vergne (1999), *inter alia*, for French.

Direkt Profil applies a cascade of three sets of rules to produce the four layers of annotations. The first unit segments the text in words. An intermediate unit identifies the prefabricated expressions. The third unit annotates simultaneously the parts of speech and the groups. Finally, the engine creates a group of results and connects them to a profile. It should be noted that the engine neither annotates all the words, nor all segments. It considers only those which are relevant for the determination of the stage. The engine applies the rules from left to right then from right to left to solve certain problems of agreement.

The current version of Direkt Profil is available online from this address: `http://www.rom.lu.se:8080/profil`. This version of the system implements phenomena related to the verb phrase. In Granfeldt et al. (2005b) the performance of Direkt Profil version 1.5.2 was evaluated. The results showed an overall F-measure of 0.83.

	NPs	VPs	PPs	Conj	MWE	Total
Reference structures	216	152	112	69	29	578
Detected structures	222	163	86	73	26	570
Correctly detected structures	208	137	85	69	26	525
Recall	96%	90%	76%	100%	90%	91%
Precision	94%	84%	99%	95%	100%	92%
F-measure	0.95	0.87	0.86	0.97	0.95	0.91

Table A.3: Results on segments. We have excluded misspelled words from the reference annotation. Taking into account all the words would probably yield lower figures.

A.5 Determining Profiles with a Machine Learning Approach

The linguistic constructions behind the profiling method are the result of systematic empirical observations and analyses of longitudinal corpora. The stages of development and the phenomena that make them up were presented in Bartning and Schlyter (2004). These are elaborated on the basis of more than 80 individual

recordings. In all, some 25 phenomena are taken into consideration when establishing a learner profile and a learner stage. In the text classification step, we consider these phenomena as features that represent the learners' texts.

A.5.1 Optimizing Feature Selection

We manually classified the texts of the subcorpus *Le voyage en Italie* (see Table A.1) according to the development stage they were reflecting. We developed a machine learning approach to optimize the profiles on the basis of this classification. Optimizing can be of at least two types. First, this approach will limit the need for manual parameter tuning. Using this technique, we expect to be able to narrow down percentage spans like those in Table A.2. For example the span for nonfinite lexical verbs at Stage 1 is estimated to go from 50% to 75%. Using this feature as a vector in the machine learning algorithm, we expect to be able to add more precision to this estimation. A second type of improvement is the identification of new features or feature engineering. In text classification, feature vectors often contain up to 10,000 features (Joachims, 1997). It is probable that we have not yet identified all the relevant features to classify learner texts according to their stage of development. Since the Direkt Profil annotation is far richer than the 25 features identified manually, there is a potential for identifying more relevant features.

Raw scores for new features can be obtained by simply counting how many times a certain rule has been applied by the analyzer. Via simple processing, we can also obtain ratios which are often better measures, for example the ratio of inflected verbs to the total number of verbs.

A.5.2 Machine Learning Algorithms and Tools

The machine-learning module uses decision trees based on the ID3 algorithm (Quinlan, 1986) and Support Vector Machines (Boser et al., 1992). The training phase automatically induces classifiers from texts in the CEFLE corpus that we manually classified and the features we extract with the analyzer. Once trained, the system uses the decision trees to automatically classify texts from the learners at runtime. We will present results for three types of classifiers: C4.5, SVM classifiers, and LMT (Landwehr et al., 2003). All our experiments have been done with the Weka collection of machine learning algorithms (`http://www.cs.waikato.ac.nz/ml/weka`).

A.5.3 The Profiler Optimization Sequence

In order to describe how we are working with profile optimization, consider first the following sample learner text from the CEFLE corpus:

Marie et Sofia est deux filles. Marie est grosse et a blonde cheveux. Sofia est mince et a marron cheveux. Elles aimaies travaillient. Sur une semaine elles sommes travaillient en Italie. Iatlie est dans le sud en Europe. Marie a une petite vert voiture. Dans la autoroute farie de la voiture sur Italie. Le temps est belle. Arrive l'hotel Marie et Sofia sortient sur votres etage dans l'hotel. La etage est petit et a une grosse venster. Prochein semain elles baigne dans la mer. Sur la soir Marie et Sofia avec deux hommes faire le disco. Il est amour dans le voyage! Un de voyage en Italie elles faire un a rote bus sur un sightseeing. Le finir en de voyage travaillent Marie,Sofia et de deux hommes "back to" Suede!

This text was written by a learner at stage 1. The text contains a number of features typical for learner texts and it can be analyzed for developmental stage using the developmental sequences in Table A.2. Here we will focus on those features that we have used in our first experiments to train the automatic classifier. These include some features from Table A.2, e.g. percentages of finite forms of lexical verbs in obligatory contexts and subject-verb agreement (all grammatical persons collapsed) but also a number of other features. In addition to grammatical features, we have used lexical features, e.g. type-token ratio (TTR), a list of all the words in the text and word frequency information. For the last feature, token frequency in a large corpus of written French, we have extracted information from the *Lexique* database (New et al. (2004); `http://www.lexique.org`). In total, 33 features were used in the training session. These are presented in Table A.4 with their respective values for this particular learner text.

Figure A.1 shows an example of a resulting decision tree for classifying learner texts according to their developmental stage. Without going into to details at this preliminary stage, it is particularly interesting that the decision tree presents the features in an hierarchical manner, following their classifying weight. This will help us in further developing the profiles and adding relevant features to them (feature engineering).

A.5.4 Evaluating Classifier Performance

We evaluated the performance of the three different classifiers used, C4.5, SVM and LMT. We carried out two separate evaluations. We first clustered the five stages into three larger stages, where stages 1 and 2, respectively 3 and 4, were collapsed into two stages. We then ran a second evaluation with the original five stages. Currently, the best classifier, SVM, obtains an average precision and recall in the vicinity of 70 % for the three-stage classification, and an average of 43 % precision and 36 % recall in the five-stage classification. As can be seen in Table A.5, the C4.5 and LMT classifiers perform less well.

Tables A.5 and A.6 show that the difficulty is to automatically discriminate between texts from neighboring stages (i.e. 1 from 2, 3 from 4, etc.). We believe

```
Finiteness - inflected and uninflected verbs <= 5
|   Inflected verbs <= 4: 1 (10.0/2.0)
|   Inflected verbs > 4: 2 (2.0)
Finiteness - inflected and uninflected verbs > 5
|   Average sentence length <= 10
|   |   TTR <= 47
|   |   |   Verbs in the conditional <= 0
|   |   |   |   Percentage lexical present tense verbs with agreement <= 60: 2 (10.0)
|   |   |   |   Percentage lexical present tense verbs  with agreement > 60
|   |   |   |   |   Lexical verbs in present tense <= 2: 2 (6.0)
|   |   |   |   |   Lexical verbs in present tense > 2
|   |   |   |   |   |   Occurrences of the 1,000 most frequent words <= 589: 2 (6.0/2.0)
|   |   |   |   |   |   Occurrences of the 1,000 most frequent words > 589: 3 (9.0)
|   |   |   Verbs in the conditional > 0: 3 (3.0)
|   |   TTR > 47: 1 (3.0/1.0)
|   Average sentence length > 10
|   |   Occurrences of the next 2,000 words <= 33
|   |   |   Word count <= 344: 3 (8.0/1.0)
|   |   |   Word count > 344
|   |   |   |   Percentage inflected verbs <= 91: 3 (2.0/1.0)
|   |   |   |   Percentage inflected verbs > 91: 4 (14.0/2.0)
|   |   Occurrences of the next 2,000 words > 33
|   |   |   Percentage participles with stem error <= 14
|   |   |   |   Lexical verbs in the present tense <= 0: 4 (3.0/1.0)
|   |   |   |   Lexical verbs in the present tense > 0
|   |   |   |   |   Sentences without verbs <= 1
|   |   |   |   |   |   Average sentence length <= 13
|   |   |   |   |   |   |   Occurrences of the 1,000 most frequent words <= 654: 3 (2.0)
|   |   |   |   |   |   |   Occurrences of the 1,000 most frequent words > 654: 6 (2.0)
|   |   |   |   |   |   Average sentence length > 13: 6 (15.0)
|   |   |   |   |   Sentences without verbs > 1
|   |   |   |   |   |   Finiteness - inflected and uninflected verbs <= 16
|   |   |   |   |   |   |   Occurrences of non-dictionary words <= 334: 2 (3.0/1.0)
|   |   |   |   |   |   |   Occurrences of non-dictionary words > 334: 3 (2.0)
|   |   |   |   |   |   Finiteness - inflected and uninflected verbs > 16: 6 (2.0)
|   |   |   Percentage participles with stem error > 14: 2 (3.0/1.0)
```

Figure A.1: A excerpt of the decision tree.

that one reason is due to the fact that Direkt Profil 1.5.2 only analyzes a subset of the phenomena described in Bartning and Schlyter (2004). Consequently, the classifying algorithm can currently not be trained with the full range of developmental sequences. We are therefore developing an enhanced, more flexible parser, which will make more features detectable, and hopefully improve classification accuracy significantly. The improved parser is near completion, and further results are expected in 2006.

A.6 Conclusion

In this paper, we have presented a new CLC in French. The CEFLE corpus (*Corpus Écrit de Français Langue Étrangère*) contains written texts in French produced by adolescent Swedish learners of French. It also contains a control group with texts written by French adolescents on the same topics. We have developed an analyzer called Direkt Profil on the basis of this CLC. The analyzer carries out a sentence analysis of learner texts based on developmental sequences. In this paper we have presented two new features of Direkt Profil and evaluated them. The first one is the introduction of a chunking layer to our annotation. In this layer the system identifies four syntactic groups. The evaluation of this annotation is presented in Table A.3. The second new feature is the introduction of a machine-learning module to optimize profiles, carry out parameter tuning and identify new features for profiling linguistic development on the basis of learner texts. We presented some initial results on classification using five different features. For a three stages classification the average precision and recall reaches 70 %. As Direkt Profil continues to develop we expect the performance of the classifier system to increase considerably within the next couple of months.

A.7 Acknowledgments

The research presented here is supported by a grant from the Swedish Research Council, grant number 2004-1674 to the first author and by grants from the Elisabeth Rausing foundation for research in the Humanities and from Erik Philip-Sörenssens foundation for research.

TTR	47
Occurrences of the 1,000 most frequent words	589
Occurrences of the next 2,000 words	13
Occurrences of less frequent words	0
Occurrences of non-dictionary words	397
Conjunctions	7
Word count	136
Number of sentences	16
Average sentence length	8
Sentences without verb	1
Finiteness: total of inflected and uninflected verbs	4
Inflected verbs	3
Lexical verbs in the present tense	1
Verbs in the passé composé	1
Modal auxiliaries + infinitives	0
Verbs in imparfait	0
Être/avoir in imparfait	0
Lexical verbs imparfait	0
Lexical verbs imparfait with agreement	0
Verbs in the simple future	0
Lexical verbs in the simple future	0
Verbs in the pluperfect	0
Verbs in the conditional	0
Percentage inflected verbs	75
Percentage inflected verbs with agreement	33.33
Percentage sentences without verb	6.25
Percentage lexical present verbs with agreement	0
Percentage verbs in passé composé with agreement	0
Percentage participles with stem error	0
Percentage simple future being être/avoir	0
Percentage lexical simple future verbs with agreement	0
Percentage lexical conditional with agreement	0
Stage	1

Table A.4: An example of feature vector.

	C4.5		SVM		LMT	
Stage	Precision	Recall	Precision	Recall	Precision	Recall
1–2	0.63	0.62	0.67	0.77	0.68	0.69
3–4	0.54	0.57	0.76	0.66	0.70	0.64
6	0.62	0.59	0.91	0.91	0.76	0.86

Table A.5: Results of the classification of texts into 3 stages for the three classifiers. Each classifier used 33 attributes and was trained on the *Voyage en Italie* corpus.

	C4.5		SVM		LMT	
Stage	Precision	Recall	Precision	Recall	Precision	Recall
1	0.50	0.40	0.57	0.40	0.44	0.40
2	0.37	0.38	0.47	0.62	0.45	0.48
3	0.23	0.25	0.43	0.36	0.46	0.39
4	0.47	0.44	0.67	0.63	0.56	0.56
6	0.62	0.59	0.91	0.91	0.76	0.86

Table A.6: Results of the classification of texts into 5 stages for the three classifiers. Each classifier used 33 attributes and was trained on the *Voyage en Italie* corpus.

Appendix B

A Quick Introduction To Weka

B.1 The Attribute-Related File Format

Weka's native file format for datasets is the Attribute-Related File Format (ARFF). It is a raw text file with one metadata (header) and one data part. The header typically looks like this:

```
@relation LexicalProfiler

@attribute 'Occurrences of the 1,000 most common words' integer
@attribute 'Occurrences of the next 2,000 words' integer
@attribute 'Occurrences of less frequent words' integer
@attribute 'Occurrences of non-dictionary words' integer
@attribute 'Classification' {1, 2, 3, 4, 6}
```

In the first line, the name of the dataset is given. If any editing to the data has been applied in the Weka GUI, those changes will be reflected on this line as well.

The header continues with a listing of the attributes. Notice that metadata is always preceded by an @. The last attribute, Classification, is somewhat different to the other ones, since it is a so-called *nominal* attribute, having a discrete and limited range[1] The other attributes are *integers*, a type of *numeric* attributes. Other attribute types present in Weka are *real*, *string*, *date* and *relational*. For the study in this report, numeric and nominal attributes are the only ones encountered. The syntax is illustrated in the example above.

[1] It is not necessary to give nominal attributes numbered values; any string works, but each value should only be used once in the declaration; so they could have been called *Beginner*, *Novice*, *Intermediate*, *Advanced* and *Native* instead.

After the header follows the data section, which typically looks like this:

```
@data
463,47,0,488,3
594,30,0,374,3
530,49,0,420,3
430,80,3,485,2
481,27,0,491,2
433,48,0,518,2
509,55,4,430,3
475,49,0,475,2
428,23,0,547,2
480,60,2,455,2
433,96,4,465,1
442,53,0,503,2
443,66,0,490,1
```

Preceded by the *@data* marker, the instances in the dataset are listed each on one comma-separated line, with attributes occurring in the same order as in the header.

B.2 Processing ARFF data

B.2.1 Opening an ARFF file

To open an ARFF file in Windows, simply double-click on it in the folder window. The Weka Explorer GUI will appear with the data loaded.

B.2.2 Defining new attributes

For defining new attributes, there is a large set of filters, which can be used. To create percentages from counter values, as have been done in the project described in the main report, the filter `weka.filters.unsupervised.attribute.AddExpression` was used. Filters are chosen with the *Choose* button in the upper-left corner. The filter name and its standard parameters are displayed in the filter field on the right of the Choose button. To edit the parameters, simply click in the filter field.

Suppose that we want to define the new attribute *Percentage Determiner-Noun in agreement*. To calculate that percentage, two counter values are needed: *Total number of Determiner-Noun occurrences* (attribute number 9; `a9`) and Number of Determiner-Noun occurrences in agreement (attribute number 10; `a10`). The formula for the percentage is `100*a10/a9`. Enter the new attribute name and the

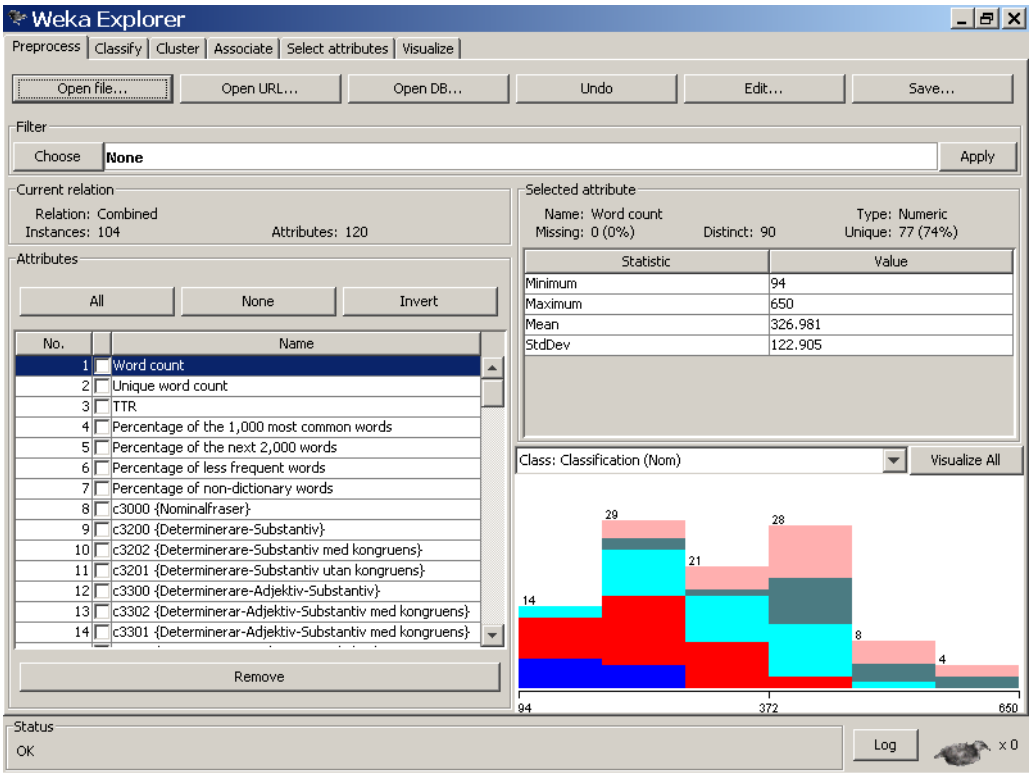

Figure B.1: The Weka main GUI

defining formula in the dialogue box as shown in Figure B.2. Finally, press *OK* to save the changes to the filter field, then press *Apply* to add the new attribute to the dataset.

B.2.3 Tidying up

After all new attributes have been added, the *Classification* attribute should be moved to the end of the attribute list, since when classifiers are tested, the final attribute will by default be used as the classification. This is accomplished by copying the *Classification* attribute in approximately the same way as the previous attributes were defined by formulae, but here the `weka.filters.unsupervised.attribute.Copy` filter is used instead, since `AddExpression` would transform the nominal values into numerical ones, which cannot be handled by the classifiers that have been used in the main report study.

Finally, the attributes from which the new percentages have been made, and the *Classification* attribute in the middle of the attribute list, should be removed. To do this, simply check the boxes on the redundant attributes, then press the *Remove* button, in the lower left corner of the Weka Explorer main GUI.

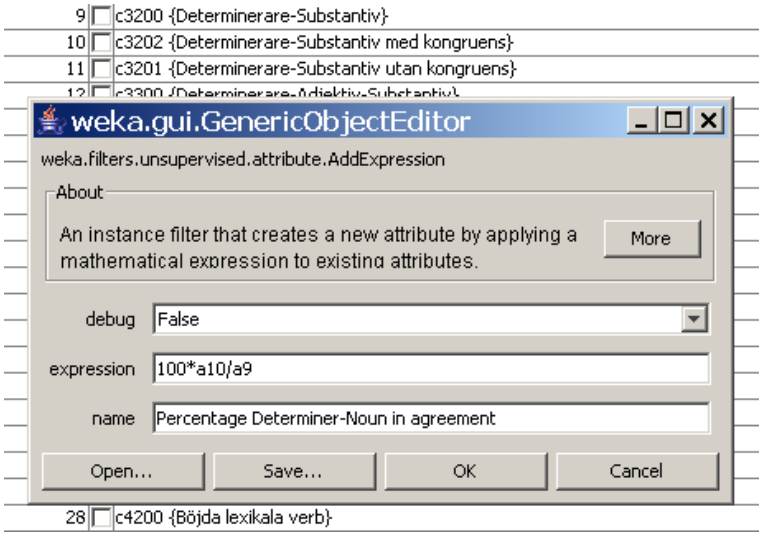

Figure B.2: The GUI for adding a new attribute

B.3 Testing classifiers

B.3.1 Setting up the test

To test classifiers, switch to the *Classify* tab in the Explorer, and then choose a type of classifier (see Figure B.3). The classifier field to the right permits adjustments of classifier parameters and is used in the same way as the filter parameter dialogue boxes described earlier, although, when the classifier parameters and the test set have been specified, the next step is to press the *Start* button, instead of applying the filter.

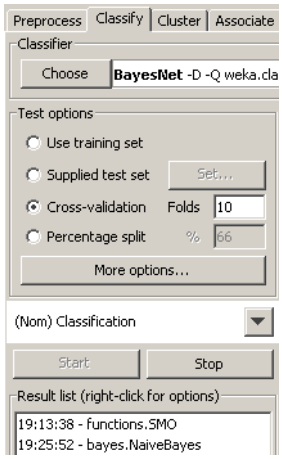

Figure B.3: The touchable part of the Classify tab

B.3.2 Evaluating the results

Here is a sample result summary from an SVM experiment with default parameters.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          66               63.4615 %
Incorrectly Classified Instances        38               36.5385 %
Kappa statistic                          0.5246
Mean absolute error                      0.2612
Root mean squared error                  0.3477
Relative absolute error                 84.1489 %
Root relative squared error             88.2881 %
Total Number of Instances              104

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.667     0.011      0.857     0.667     0.75     1
  0.655     0.147      0.633     0.655     0.644    2
  0.607     0.211      0.515     0.607     0.557    3
  0.563     0.091      0.529     0.563     0.545    4
  0.682     0.024      0.882     0.682     0.769    6

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
  6  3  0  0  0 |  a = 1
  1 19  9  0  0 |  b = 2
  0  6 17  4  1 |  c = 3
  0  1  5  9  1 |  d = 4
  0  1  2  4 15 |  e = 6
```

The confusion matrix above shows that for level 3 (intermediate) 17 texts have been correctly classified, whereas 6 have been classified as level 2 (novice), 4 have been classified as level 4 (pre-advanced), and one has been classified as level 6 (native). It is not neccessary that the classification model is inadequate, since fewer than two thirds of the texts at level 3 have been correctly classified; if those texts were given to several human assessors, who worked independently, probably not all of them would place every text at level 3.

B.4 Saving classifiers

When adequate performance is achieved, it is time to save the classifier. To do this, choose the appropriate model amongst the recently tested ones, right-click on it, then select *Save model as*. After the classification model has been saved to a file, it can be loaded and used by any program that links to the Weka libraries or is designed to use the Weka model format natively.

Bibliography

Ågren, M. (2005). Le marquage morphologique du nombre dans la phrase nominale. une étude sur l'acquisition du français L2 écrit. Technical report, Institut d'études romanes de Lund. Lund University, Lund.

Bailey, N., Madden, C., and Krashen, S. (1974). Is there a 'natural sequence' in adult second language learning. *Language Learning*, 24(2):235–243.

Bartning, I. and Schlyter, S. (2004). Stades et itinéraires acquisitionnels des apprenants suédophones en français L2. *Journal of French Language Studies*, 14(3):281–299.

Blom, G. and Holmquist, B. (1998). *Statistikteori med tillämpningar*. Studentlitteratur, Lund, Sweden, 3rd edition.

Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh. ACM.

Clahsen, H. (1986). *Die Profilanalyse*. Springer-Verlag, Berlin, Germany.

Council of Europe (2001). Common European Framework of Reference for Languages: Learning, teaching, assessment.

Dulay, H. C. and Burt, M. K. (1974). Natural sequences in child second language acquisition. *Language Learning*, 24:37–53.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.

Gendner, V., Vilnat, A., Monceaux, L., Paroubek, P., and Robba, I. (2004). Les annotations syntaxiques de référence PEAS. Technical report, LIMSI, Orsay. http://www.limsi.fr/Recherche/CORVAL/easy/PEAS_reference_annotations_v1.6.html.

Granfeldt, J. (2003). *L'acquisition des catégories fonctionelles*. PhD thesis, Lund University.

Granfeldt, J., Nugues, P., Ågren, M., Thulin, J., Persson, E., and Schlyter, S. (2006). CEFLE and Direkt Profil: A new computer learner corpus in French L2

and a system for grammatical profiling. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 565–570, Genoa, Italy.

Granfeldt, J., Nugues, P., Persson, E., Persson, L., Kostadinov, F., Ågren, M., and Schlyter, S. (2005a). Direkt Profil : un système d'évaluation de textes d'élèves de français langue étrangère fondé sur les itinéraires d'acquisition. In *Actes de la conférence Traitement Automatique des Langues Naturelles, TALN & RECITAL 2005*, volume Tome 1 – Conférences principales, pages 113–122, Dourdan, France.

Granfeldt, J., Nugues, P., Persson, E., Persson, L., Kostadinov, F., Ågren, M., and Schlyter, S. (2005b). Direkt Profil: A system for evaluating texts of second language learners of French based on developmental sequences. In *Proceedings of The Second Workshop on Building Educational Applications Using Natural Language Processing, 43rd Annual Meeting of the Association of Computational Linguistics*, pages 53–60, Ann Arbor.

Granger, S. (2004). Practical applications of learner corpora. In Lewandowska-Tomaszczyk, B., editor, *Practical Applications in Language and Computers (PALC 2003)*, pages 291–301. Peter Lang, Frankfurt.

Ågren, M. (2005). *Développement de la morphologie du nombre en français langue étrangère à l'écrit. Étude transversale.* Licentiate thesis, Lund University.

Joachims, T. (1997). Text categorization with support vector machines: Learning with many relevant features. Technical Report LS-8 Report 23, Universität Dortmund.

Kostadinov, F. (2005). Direkt Profil – Implementation of a text critiquing system for non-native students of French. Master's thesis, University of Zurich.

Kostadinov, F. and Thulin, J. (2003). A text critiquing system for Swedish-speaking students of French. Technical report, Department of Computer Science. Lund Institute of Technology.

Landwehr, N., Hall, M., and Frank, E. (2003). Logistic Model Trees. In Lavrac, N., Gamberger, D., Todorovski, L., and Blockeel, H., editors, *Proceedings of the 14th European Conference on Machine Learning (ECML)*, volume 2837 of *Lecture Notes in Computer Science*, pages 241–252. Springer.

Lundgren, J., Rönnkqvist, M., and Värbrand, P. (2003). *Optimeringslära.* Studentlitteratur, Lund, Sweden.

Malvern, D., Richards, B., Chipere, N., and Durán, P. (2004). *Lexical Diversity and Language Development.* Palgrave MacMillan, Basingstoke.

New, B., Pallier, C., Brysbalert, M., and Ferrand, L. (2004). Lexique 2: A new French lexical database. *Behavior Research Methods, Instruments, and Computers*, 36(3):516–524.

Osuna, E., Freund, R., and Girosi, F. (1997). An improved training algorithm for support vector machines. In *Proceedings of IEEE NNSP '97*, pages 53–60.

Persson, L. (2004). Direkt Profil – Ett verktyg för morfologisk analys av skriven inlärarfranska. Master's thesis, Lund University.

Pienemann, M. (1998). *Language Processing and Second Language Development.* Benjamins, Amsterdam.

Platt, J. C. (1998). Sequential Minimal Optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research, Redmond, WA, USA.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Quinlan, J. R. (1993). C4.5: Programs for machine learning.

Schlyter, S. (2003). Stades de développement en français L2. Technical report, Institut d'études romanes de Lund, Lund University. Online.

Selinker, L. (1972). Interlanguage. *International Review of Applied Linguistics*, 10(3):209–231.

Vergne, J. (1999). *Étude et modélisation de la syntaxe des langues à l'aide de l'ordinateur. Analyse syntaxique automatique non combinatoire. Synthèse et Résultats.* Habilitation à diriger des recherches, Université de Caen.

Wikipedia (2006). Naive Bayes classifier.

Witten, I. and Frank, E. (2005). *Data Mining : Practical Machine Learning Tools and Techniques.* Elsevier, 2nd edition.