Freshmen's CL curriculum: the benefits of redundancy *

Heike Zinsmeister
Department of Linguistics
University of Konstanz
78457 Konstanz, Germany
Heike.Zinsmeister@uni-konstanz.de

Abstract

In the course of the European Bologna accord on higher education, German universities have been reorganizing their traditional "Magister" and "Diplom" studies into modularized bachelor's and master's programs. This revision provides a chance to update the programs. In this paper we introduce the curriculum of a first semester B.A. program in Computational Linguistics which was taught for the first time last semester. In addition, we analyze the syllabi of four mandatory courses of the first semester to identify overlapping content which led to redundancies. We suggest for future semesters to reorganize the schedules in a way that students encounter recurring topics iteratively in a constructive way.

1 Introduction

We present the first semester curriculum of a newly revised bachelor's program in Computational Linguistics at the University of Heidelberg, Germany, which was taught for the first time at the Department of Computational Linguistics last winter semester. Four courses are mandatory for the students in the first semester: a comprehensive *Introduction to Computational Linguistics*, backed up with a course on *Formal Foundations* that emphasizes mathematical topics, and a general introduction to linguistic core modules in *Foundations of Linguistic Analysis*, the set up is completed by an *Introduction to Programming* that introduces core concepts of programming employing the programming language Python.

The parallel design leads to a situation in which related topics are introduced in the same semester in parallel fashion. Redundant duplication per se is to be avoided given that lecture time is always too sparse and should be used most efficiently such that there is enough room for examples, short in-course exercises, questions and discussions.

We analyzed the syllabi for common topics and plotted these topics to see whether they are dealt with in a constructive way across the curriculum. For future semesters we suggest some reorganization to optimize the courses' interactions. Since all courses are taught in the department of Computational Linguistics, decisions on both the courses' subtopics as well as their temporal sequencing is in full control of the local department.

We think that it is reasonable to keep the common topics and even the redundancy of introducing them in more than one course only. Iterative reintroduction could be helpful for the students if it is accompanied by a reference to the earlier mention as well as a motivation of the specific relevance for the course at hand. We expect that such an iterative approach reinforces understanding since it allows the students to build upon their prior knowledge and, furthermore, to approach the very same concept from different perspectives. This iterative method is inspired by the idea of *spiral learning* in the sense of Jerome S. Bruner (Bruner, 1960) which builds on a constructivist view on learning. It assumes that learning is an active process in which learners construct new ideas or concepts based upon

*This paper is about the curriculum taught at the Department of Computational Linguistics at the University of Heidelberg, where the author used to work.

Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics (TeachCL-08), pages 19–26,
Columbus, Ohio, USA, June 2008.

their prior knowledge. A curriculum can support this process if it revisits its basic ideas repeatedly: "the spiral curriculum [..] turns back on itself at higher levels" (Bruner, 1960, p.53).

The rest of this paper is organized as follows. First, we briefly sketch the Bologna Process, an effort of harmonizing higher education in Europe and also the special situation in Heidelberg being the background against which the bachelor's program described is created. Then, we introduce the bachelor's program of Computational Linguistics at the University of Heidelberg in Germany and describe its four mandatory courses of the first semester. We analyze the syllabi for common topics, and, finally, present a re-organized schedule for future semesters which is inspired by an iterative learning approach.

2 Background

The European Bologna Process is an effort of European countries to establish a common higher education area by the year 2010. Its central element is the introduction of a two-cycle study system consisting of bachelor's and master's degrees with comparable qualifications throughout Europe based on a common credit transfer system which allows for comparing the workload of individual courses.[1]

In the course of this international harmonizing effort, German universities are reorganizing their programs from traditional "Magister" or "Diplom" programs to modular bachelor's and master's programs. Previously "Magister" or "Diplom" was first degree in Germany, i.e. a bachelor's degree did not exist. A characteristic of the traditional programs was the freedom of choice they offered to their students, more pronounced in the "Magister" programs than in the "Diplom" programs the latter of which were traditionally realized in more technically oriented disciplines and the former in the humanities. Both type of programs were set up with a standard period of study of nine to ten semesters but the average student required more than this. European bachelor's programs predetermine a highly structured curriculum and offer a first university degree after six or seven semester of study.

The Computational Linguistics department in Heidelberg was pioneering with an early bachelor's program devised by Peter Hellwig at the beginning of the Bologna Process. Adaptions of the original bachelor's program became necessary due to general developments in the international Bologna policy and finally the need for a revised program emerged. This was realized in 2007 by Anette Frank who had filled the by then vacant chair in Computational Linguistics. The change of the departmental head brought a change from a more vocationally oriented program that prepared students to take jobs in the local language technology industry to a more academically oriented one, which is reflected in the revised syllabus. We will point to differences between the original program and the revised program where relevant.

3 The Study of Computational Linguistics in Heidelberg

> Computational linguistics (CL) is a discipline between linguistics and computer science which is concerned with the computational aspects of the human language faculty. [...] The applied component of CL is more interested in the practical outcome of modeling human language use. The goal is to create software products that have some knowledge of human language. [...] Theoretical CL takes up issues in formal theories. [...] Computational linguists develop formal models simulating aspects of the human language faculty and implement them as computer programs. (www.aclweb.org/nlpfaq.txt, credited to Hans Uszkoreit)

This quote from Hans Uszkoreit outlines the knowledge and skills that a study of CL should equip its students with: programming skills, handling of formal models, algorithmic thinking and last but not least an explicit knowledge of linguistic analysis.

All four areas are covered in our freshmen's classes which are introduced in more detail in subsequent subsections after the presentation of the overall program.

[1] One European Credit Transfer System point corresponds to 25-30 hours workload on the student cf. http://www.uni-heidelberg.de/studium/bologna/materialien/diploma/ECTSUsersGuide05.pdf. For the Bologna Process in general see http://ec.europa.eu/education/policies/educ/bologna/bologna_en.html.

In Heidelberg, B.A. students have to collect 180 credit points to complete their study. They normally enroll in two or three subjects which means that they take Computational Linguistics as main subject (in which it provides 75% of the overall workload), secondary main subject (50%) or minor subject (25%) [2] in combination with complementary subjects in the areas of computer science[3], humanities, psychology, economics, or law. Table 1 gives an overview of the courses in a 75% B.A. The first semester requirements are the same in all B.A. options involving Computational Linguistics.[4] In addition to the courses depicted in Table 1 students need to gain credits in *Generic Competences* ('übergreifende Kompetenzen' aka soft skills and courses from other departments of the faculty).[5]

3.1 The Curriculum

We thought it relevant for the students to get acquainted with Computational Linguistics proper as early as the first semester. Therefore, in addition to an introduction to formal foundations and programming a comprehensive introduction to algorithms and analysis in computational linguistics is mandatory. It was the first time that this combination of courses was taught. Before that, the Introduction to Computational Linguistics also introduced students to core linguistic topics which were spread across the whole course. The motivation for an independent introduction to linguistics was that students should get a profound background knowledge in linguistic analysis such that further courses could build on them. Before that, even basic concepts such as *morpheme* had to be reintroduced. Introduction to Programming and Formal Foundations used to be in complementary distribution due to the fact that they used to be taught by the one and the same person. An additional lecturer position in the department allowed us to to offer both courses in parallel.

The Freshmen's curriculum consists of four mandatory courses which are described in the following.

3.1.1 Introduction to Computational Linguistics

The core lecture of the first semester is the Introduction to Computational Linguistics. It is held four hours a week and is worth six credit points. It introduces the foundations of Computational Linguistics, its research objectives and research methods. It provides an overall survey of the field: the levels of language description, formal-mathematical and logical models as well as algorithmic approaches for processing such formal models. Specific topics are: dealing with ambiguities, approximation of linguistic regularities, and the relation of language and knowledge; some applications of Computational Linguistics are also introduced. Mandatory readings are selected sections from Jurafsky & Martin (2000), complemented by chapters from Carstensen et al. (2004) and Bird et al. (forthcoming).

This course is seen as the backbone of the first semester curriculum. We therefore list the lectures in detail. The content of the other three courses is only briefly described below and will be discussed in Section 4.

The first part of the schedule was strongly inspired by Jurafsky & Martin (2000):

- Sub-token level (3 lectures): computing morphology by means of regular expressions, automata, and transducers.
- Token level and context (4 lectures): identifying tokens and computing them by means of tokenizing, edit distance, n-grams, and part-of-speech tagging.
- Syntactic level (6 lectures): syntactic analysis in terms of constituency, dependency, phrase structure grammars and probabilistic context free grammars; formal grammar types: computation of syntactic structure by means of parsing strategies and parsing algorithms, and syntactic resources in terms of treebanks.

The second part of the schedule built more on Carstensen et al. (2004). It mainly dealt with semantic issues in term of analysis, computation, and resources.

[2] The minor subject option had to be introduced due to formal requirements. It is likely to be dispensed with in the future.

[3] Computer science can only be taken as minor subject.

[4] In the 25% B.A. the workload on students is reduced. They only need to attend one of the two courses on formal foundations either *Mathematical Foundations* in the first semester or *Logical Foundations* in the second one.

[5] In the 75% B.A. students need to collect 20 credit points in *Generic Competences* during their three-year study.

<table>
<tr><th>Semester</th><th colspan="2">Computational Linguistics Modules</th><th>Linguistic Modules</th><th colspan="2">Computational Modules</th></tr>
<tr><td>6</td><td colspan="5">BA-Thesis, Oral Exam</td></tr>
<tr><td>5</td><td colspan="3" rowspan="2">Advanced Studies (Computational Linguistics or Formal Linguistics)
Core Studies in Computational Linguistics</td><td rowspan="3">Core Studies in Theoretical or Applied Computer Science</td><td rowspan="3">Software Project</td></tr>
<tr><td>4</td></tr>
<tr><td>3</td><td>Statistical Methods for CL</td><td>Algorithmic CL</td><td>Formal Semantics</td></tr>
<tr><td>2</td><td></td><td>Logical Foundations</td><td>Formal Syntax</td><td colspan="2">Advanced Programming</td></tr>
<tr><td>1</td><td>Introduction to CL</td><td>Mathematical Foundations</td><td>Foundations of Linguistic Analysis</td><td colspan="2">Introduction to Programming</td></tr>
</table>

Table 1: Modules in B.A. Computational Linguistics (75%)

- predicate logic (2 lectures)
- propositional logic and inferences (2 lectures)
- compositional semantics and Lambda calculus (1 lecture)
- lexical semantics including resources (2 lectures)
- discourse semantics / pragmatics (1 lecture)

The schedule was rounded off by two lectures on applications, in particular information extraction and machine translation.

There were eight assessments during the semester of which students had to pass 60%. Most of them dealt with theoretical comprehension, two more practical assessments involved an introduction to basic UNIX tools, and (probabilistic) parsing with the NLTK tools (Bird et al., forthcoming). We decided to split the written exam into two sub-exams, the first one took place in half time the second one in the final week of the semester. Thus students could better focus on the topics at hand.

3.1.2 Formal Foundations part 1: Mathematical Foundations

Formal Foundations is held two hours a week and is worth six credit points. The theory of formal languages is a prerequisite for e.g. model-theoretic semantics and parsing approaches. This lecture in particular deals with mathematical foundations, formal languages and formal grammars, regular expressions and finite automata, context-free languages, context-sensitive languages and Type-0 languages, Turing machines, and computability theory. The recommended reading includes Schöning (2001), Klabunde (1998), Partee et al. (1990), as well as Hopcroft and Ullman (1979).

There were eight graded assessments and the students had to pass 50% of the overall tasks .

3.1.3 Foundations of Linguistic Analysis

The introduction to linguistics is also held two hours a week and is worth four credit points. Linguistic knowledge is a distinctive property of computational linguistics. In this lecture students get a thorough introduction to the core modules of the language faculty: phonetics and phonology, morphology, syntax, semantics, and pragmatics with a special emphasis on linguistic phenomena of German. The core reading was Meibauer et al. (2002).

There were ten small assessments of which the students had to pass eight.

3.1.4 Introduction to Programming

The fourth mandatory course is held four hours a week and is worth six credit points. In this lecture, students learn to devise algorithmic solutions and implementations for problems related to Natural Language Processing. Moreover, the course introduces basic principles of software engineering in

order to equip the students with skills to develop correct and maintainable programs. These capabilities are further facilitated in the Advanced Programming course during the second semester and a comprehensive hands-on software project during the advanced phase of undergraduate studies.

Recommended reading is Demleitner (unpublished), Lutz and Ascher (2007), Martelli (2006), as well as the official Python documentation (van Rossum, 2008).

There were ten programming assessments of which the students had to hand in eight and earn half of the points to be permitted to take the final exam.

3.2 Local Conditions

3.2.1 Students

Students require higher education entrance qualification and no other prerequisites. Language of instruction is German but students come from various countries and speak a diversity of native languages, including Bulgarian, Chinese, English, French, Italian, Japanese, Kurdish, Polish, Russian, Spanish, Turkish, Turkmen and Ukrainian. About 40 students enrolled in Computational Linguistics, about two third of which classified themselves as programming beginners. In general about 20% of the first semester students failed at least one of the courses first time.

3.2.2 Realization of Courses

Three of the four courses under examination are taught by faculty members holding a PhD (or a comparable doctoral degree) and one by a member of the faculty still completing his doctorate. The courses are taught as lectures which are accompanied by optional tutorial sessions. These tutorials were coached by undergraduate student tutors who mainly corrected and discussed the students' assessments. The students had to hand in assessments on a regular basis which could either be solved as a group or individually depending on the course. Passing a substantial portion of the exercises was a prerequisite for being permitted to take the courses' exams. Each course provided its own wiki platform for the students to communicate easily among themselves as well as with student tutors and lecturers. The wikis were also a common platform for publishing example solutions by the tutors and keeping records of answers to students' questions.

4 Analysis of the Syllabi

The individual courses were planned in accordance with the sequence of topics in standard textbooks such as Jurafsky and Martin (2000) and Carstensen et al. (2004) for Introduction to Computational Linguistics, Schöning (2001) for Formal Foundations, and Meibauer et al. (2002) for Foundations of Linguistic Analysis. In Introduction to Programming we used a hands-on manuscript (Demleitner, unpublished).

The following list summarizes the main topics that are dealt with in more than one syllabus. Common topics include:

- modules of linguistics: ICL, FLA
- regular expressions: ICL, FF, IP
- automata: ICL, FF
- grammar types: ICL, FF
- morphology: ICL, FLA
- segmentation, tokenization: ICL, FLA, IP
- n-grams: ICL, IP
- phrase-structure grammars: ICL, FF, FLA
- parsing: ICL, FF, IP
- lexical semantics: ICL, FLA
- model in semantics: ICL, FF
- discourse semantics, pragmatics: ICL, FLA

Before the semester started, the group of lecturers met and arranged the general schedules of the courses. During the semester, the lecturers happened to lose track of the progression of other courses. In some cases explicit cross-references were given, for example in the case of lexical semantics, but most of the time, concepts were (re-)introduced in each course independently. Sometimes lecturers asked students whether they were already familiar with a newly introduced topic from other courses; then there was a short discussion in class and students

were reminded of previous mentions of that topic. In general, the didactics of the individual courses were not adapted to take account of such recurrence of topics across the curriculum.

Nevertheless, the parallel fashion of the four courses at hand seemed to be reasonable even in this form. Students deemed the interdependence between the courses as appropriate in the final evaluation of the courses. They gave it an average score of 2.052 with a standard deviation of 1.05 on a scale of 1 (very appropriate) to 6 (non-existent).

Our conclusion is that a slight rescheduling of the courses would improve teaching efficiency in the sense that lecturers could count on already introduced materials and students could benefit from recurring topics by exploring them in the context of different disciplines. Table 2 depicts our proposed schedule.

An important and easily realizable change that we suggest is to ensure that all linguistic modules are dealt with first in Foundation of Linguistic Analysis (FLA) before they are set into a more formal and also computational setting in the Introduction to Computational Linguistics (ICL). This could be realized by starting FLA with morphology right from the beginning, instead of introducing the linguistic modules first which was also part of the introduction in ICL. FLA also entered the areas of lexicography and psycho linguistics (aka the mental lexicon) which could be skipped in future semesters. Lectures on phonetics and phonology which were taught after morphology could be rescheduled to the end of the semester. Both topics are relevant for applications which were introduced in the final sessions of ICL and also for subsequent optional seminars in speech generation or speech synthesis in higher semesters.

In Formal Foundations (FF) lectures on grammars, the Chomsky hierarchy, and decision theory took place in lectures 5 and 6. They could be postponed and lectures on automata moved forward instead. This would ensure that both of these topics are dealt with in FF after they have been introduced in ICL. Formal Foundations provides a more formal and deepened insight into these topics and should, therefore, be encountered last.

In Introduction to Programming (IP) issues of algorithms and analysis are a means to an end: they are used in programming examples and assessments. Therefore, such topics should be referred to in IP only after they have been introduced in ICL. The coordination of this already worked out well with respect to n-grams and phrase structure grammars. Lectures on segmentation and regular expressions took place in the last third of the semester and could be moved forward to have them closer to their introduction in the other courses.

From a student's perspective these changes would result in a kind of spiral curriculum. For example, the first encounter with constituency and syntactic phrase structure would be in FLA, the course which is least formal and relates most to secondary school knowledge. Their second involvement with phrase structure would be in ICL and was more formal and also involved computational aspects of syntactic analysis. Then, they would learn more on the formal characteristics of grammars in FF, and finally, they perceived it as an application in an IP programming task. If these lectures are seen as stages on a common pathway of learning then they conform to the idea of spiral learning: in course of time the students return to the same concepts each time on a more advanced level.

Table 2 gives a contrastive overview of the four course curricula and shows how the individual topics could temporally related to one another to support an iterative leaning approach.

The first column counts the semester's teaching units in the average winter semester (which includes some public holidays). Introduction to Computational Linguistics (ICL) and Introduction to Programming (IP) took place twice a week, Foundations of Linguistic Analysis (FLA) and Formal Foundations (FF) only once. The 25th session is followed by another week of revision and final exams, which is not included here.

5 Conclusion

We proposed an enhanced curriculum for teaching parallel freshman's courses in Computational Linguistics, in the spirit of the newly revised bachelor's program in Computational Linguistics at the University of Heidelberg. In particular, we examined the first semester curriculum of four mandatory courses: Introduction to Computational Linguis-

#	**Introduction to Computational Linguistics**	**Formal Foundations**	**Foundations of Linguistic Analysis**	**Introduction to Programming**
1		sets, iterations, relations		introduction
2	introduction to Computational Linguistics and linguistic modules		morphology: morphemes inflection, derivation	data types
3	regular expression and automata	equivalence relation function, induction formal languages		functions and methods
4	morphology and finite automata		syntax: PoS, topological fields	strings, data structures, control structures
5	morphology and finite transducers	automata: DFAs and NFAs		sequences
6	tokenizer and spelling editor	NFAs, regular grammars regular expression		data structures: dictionaries
7	tokenizing and n-grams		syntax: phrases chunks, X-bar schema	encodings
8	tagging: rule-based, HMMs, Brill	Pumping lemma, minimizing of automata		modules, packages, tests
9	tagging		syntax: valency, semantic roles, gram. functions	modules
10	syntax and CFGs constituency, dependency	closures		exercise: n-grams
11	grammar types, parsing		syntax: sentential level CP/IP structures	regular expressions
12	parsing: bottom up, top down	grammars, left-right derivation, Chomsky hierarchy		regular expressions
13	parsing: Earley algorithm		semantics: meaning, lexical semantics	PS grammar, recursion
14	midterm exam	decision theory		file handling
15	treebanks and PCFCs	parsing: CYK algorithm		tuple, list comprehensions
16	treebanks: resources		semantics: compositional semantics	object-oriented programming: basics
17	semantics: predicate logic	pushdown automata Turing machines, computability theory		oo programming: techniques
18	Christmas puzzle: predicate logic and model theory		pragmatics: deixis, anaphora, information structure	Christmas lecture
19	semantics: propositional logic and inferences	revision: Pumping lemma		oo programming: techniques
20	semantics: propositional logic and inference		pragmatics: speech acts conversational maxims, presuppositions	exercise: segmentation
21	semantics: compositional semantics and λ-calculus	a simple grammar for English		factory functions
22	semantics: lexical semantics		phonetics	blocks and visibility
23	semantics: lexical semantics	revision		exceptions
24	semantics: discourse semantics		phonology	object customization
25	applications	exam	revision	exam

Table 2: Re-organized curriculum of first semester courses

tics, Formal Foundations, Foundations of Linguistic Analysis, and Introduction to Programming, and identified common topics. When the four courses were first held in parallel last semester, it happened that recurring topics were introduced independently without taking into account their previous mention in other courses. For future semesters we suggest a better alignment of recurring topics and sketch rearrangements of the courses' schedules. Instead of pruning recurrent topics, we think that from the perspective of the psychology of learning it is useful for the students if the same concepts and ideas are approached from different angles iteratively.

Acknowledgments

We are indebted to our co-instructors in Heidelberg: Anette Frank, teaching the Introduction to Computational Linguistics, Philipp Cimiano teaching Formal Foundations, as well as Matthias Hartung and Wolodja Wentland, co-teaching Introduction to Programming, for sharing their experiences and commenting on versions of this paper. We would also like to thank Anke Holler for valuable input on the history of the Heidelberg B.A. program, Karin Thumser-Dauth for pointing us to the work of Jerome Bruner, Piklu Gupta for commenting on a pre-final version and also for help with the English. A special thank goes to three anonymous reviewers for their very detailed and constructive comments.

References

Steven Bird, Ewan Klein, and Edward Loper. forthcoming. *Natural Language Processing in Python*.

Jerome S. Bruner. 1960. *The Process of Education*. Harvard University Press, Cambridge, Mass.

Kai-Uwe Carstensen, Christian Ebert, Cornelia Endriss, Susanne Jekat, Ralf Klabunde, Hagen Langer. eds. 2004. *Computerlinguistik und Sprachtechnologie. Eine Einführung.* Spektrum, Akademischer Verlag, Heidelberg.

Markus Demleitner. unpublished. Programmieren I. `www.cl.uni-heidelberg.de/kurs/skripte/prog1/html/`

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation* Addison Wesley.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall Series in Artificial Intelligence. Prentice Hall.

Ralf Klabunde. 1998. *Formale Grundlagen der Linguistik* Narr, Tübingen.

Mark Lutz and David Ascher. 2007. *Learning Python.* O'Reilly, 2nd Edition.

Alex Martelli. 2006. *Python in a Nutshell. A Desktop Quick Reference.* O'Reilly, 2nd Edition.

Jörg Meibauer et al. eds. 2007. *Einführung in die germanistische Linguistik*. Metzler, Stuttgart.

Barbara Partee et al.. 1990. *Mathematical Methods in Linguistics*. Kluwer, Dordrecht.

Guido van Rossum. 2008. *Python Tutorial*. Python Software Foundation. `docs.python.org/tut/tut.html`

Uwe Schöning. 2001. *Theoretische Informatik kurzgefasst*. Spektrum Akademischer Verlag in Elsevier.