ESSLLI 2010: Resource-light Morpho-syntactic Analysis of Highly Inflected Languages

Instructors: Anna Feldman & Jirka Hana

August 9-13, 2010

Contents

Chapter 1

Introduction

Simplifying somewhat, **morphological analysis** (MA) is the process of labeling a word with tags encoding the word's morphological properties. For example, the English *her* could be assigned two tags:

1. personal pronoun in object form (*I saw her.*) and
2. possessive pronoun (*I saw her son.*).

Morphological analysis considers words out of context; **morphological tagging**, on the other hand, assigns each word a single tag based on the context the word is in. Therefore, *her* would be tagged with the personal pronoun tag in the sentence *I saw her* and with the possessive tag in the sentence *I saw her son.* Depending on the language and captured distinctions, a **tagset**, a set of possible tags, usually contains between several dozens to several thousands tags. For example, there are about 40 tags in the English Penn Treebank tagset (Marcus et al. 1993), about 4,000 in the Czech Positional tagset (Hajič 2004). Morphological analysis and tagging may be accompanied by **lemmatization**, a procedure that assigns each word its lemma (also called lexeme or base form). For example, *her* could be assigned the lemma *she* or *her*.

Morphological analysis, tagging and lemmatization are essential for many Natural Language Processing (NLP) applications, of both practical and theoretical nature. They are commonly used in syntactic parsers, grammar checkers, speech recognition systems, web searches, machine translation, text-to-speech synthesis, etc.

Modern taggers and analyzers are very accurate. However, the standard way to create them for a particular language requires substantial amount of expertise, time and money. A tagger is usually trained on a large corpus (around 100,000+ words) annotated with correct tags. Morphological analyzers usually rely on large manually created lexicons. For example, the Czech analyzer by Hajič (2004) uses a lexicon with 300,000+ entries. As a result, most of the world languages and dialects have no realistic prospect for morphological taggers or analyzers created in this way.

Various techniques have been suggested to overcome this problem. The most commonly used approaches are: (1) Unsupervised methods, (2) projection of information from one language to another via parallel corpora (the same text in two or more languages). However these approaches have drawbacks that often make them hard to use in practice. The accuracy of systems developed by unsupervised or bootstrapping methods is still below the accuracy of the supervised systems. Parallel corpora are rather rare. They are also restricted to rather unnatural genres, such as the proceedings of the Canadian or European parliaments.

Chapter 2

Basics of morphology

2.1 What is morphology?

Morphology is the study of the *internal structure of words.*

- The first linguists were primarily morphologists.
- Well-structured lists of morphological forms of Sumerian words were attested on clay tablets from Ancient Mesopotamia and date from around 1600 BCE; e.g.,

badu	'he goes away'	*ing̃en*	'he went'
baddun	'I go away'	*ing̃enen*	'I went'
bašidu	'he goes away to him'	*inšig̃en*	'he went to him'
bašiduun	'I go away to him'	*inšig̃enen*	'I went to him'

(Jacobsen 1974: 53-4)

- Morphology was also prominent in the writings of Pāṇini (5th century BCE), and in the Greek and Roman grammatical tradition.
- Until the 19th century, Western linguists often thought of grammar as consisting primarily of rules determining word structure (because Greek and Latin, the classical languages had fairly rich morphological patterns).

2.2 Some terminology

Words, forms, lemmas, lexemes

There are two rather different notions of 'word': the word as it occurs in running speech or text and a word as it is listed in a dictionary:

- **Word-form**, **form**: A concrete word as it occurs in real speech or text. For our purposes, word is a string of characters separated by spaces in writing.
- **Lemma**: A distinguished form from a set of morphologically related forms, chosen by convention (e.g., nominative singular for nouns, infinitive for verbs) to represent that set. Also called the canonical/base/dictionary/citation form. For every form, there is a corresponding lemma.

- **Lexeme**: An abstract entity, a dictionary word; it can be thought of as a set of word-forms. Every form belongs to one lexeme, referred to by its lemma.

  For example, in English, *steal*, *stole*, *steals*, *stealing* are forms of the same lexeme STEAL; *steal* is traditionally used as the lemma denoting this lexeme.

- **Paradigm**: The set of word-forms that belong to a single lexeme. We will get back to this notion later in the course. As an example, the paradigm of the Latin noun lexeme INSULA 'island' is given below:

(1) The paradigm of the Latin INSULA 'island'

	singular	plural
nominative	*insula*	*insulae*
accusative	*insulam*	*insulas*
genitive	*insulae*	*insularum*
dative	*insulae*	*insulis*
ablative	*insula*	*insulis*

Complications with terminology. The terminology is not universally accepted, for example:

- lemma and lexeme are often used interchangeably
- sometimes lemma is used to denote all forms related by derivation (see below).
- Paradigm can stand for the following:
  1. Set of forms of one lexeme
  2. A particular way of inflecting a class of lexemes (e.g. plural is formed by adding *-s*).
  3. Mixture of the previous two: Set of forms of an arbitrary chosen lexeme, showing the way a certain set of lexemes is inflected.

Note: In our further discussion, we use lemma and lexeme interchangeably, and we use them both as an arbitrary chosen representative form standing for forms related by the same paradigm.

Morpheme, Morph, Allomorph

- **Morphemes** are the smallest meaningful constituents of words; e.g., in *books*, both the suffix *-s* and the root *book* represent a morpheme. Words are composed of morphemes (one or more).

  sing-er-s, home-work, moon-light, un-kind-ly, talk-s, ten-th, flipp-ed, de-nation-al-iz-ation

- **Morph**. The term morpheme is used both to refer to an abstract entity and its concrete realization(s) in speech or writing. When it is needed to maintain the signified and signifier distinction, the term **morph** is used to refer to the concrete entity, while the term morpheme is reserved for the abstract entity only.

- **Allomorphs** are variants of the same morpheme, i.e., morphs corresponding to the same morpheme, they have the same function but different forms. Unlike the synonyms they usually cannot be replaced one by the other.

(2) a. indefinite article: *an orange* – *a building*
b. plural morpheme: *cat-s* [s] – *dog-s* [z] – *judg-es* [əz]
c. opposite: *un-happy* – *in-comprehensive* – *im-possible* – *ir-rational*

The order of morphemes/morphs matters:
talk-ed ≠ **ed-talk*, *re-write* ≠ **write-re*, *un-kind-ly* ≠ **kind-un-ly*

It is not always obvious how to separate a word into morphemes. For example, consider the *cranberry*-type morphemes. These are a type of bound morphemes that cannot be assigned a meaning or a grammatical function. The *cran* is unrelated to the etymology of the word *cranberry* (*crane* (the bird) + *berry*). Similarly, *mul* exists only in *mulberry.* Other complications, namely zero-morphemes and empty morphemes, are mentioned below (see §2.6).

Bound × Free Morphemes

- **Bound** – cannot appear as a word by itself.
  -s (*dog-s*), *-ly* (*quick-ly*), *-ed* (*walk-ed*)
- **Free** – can appear as a word by itself; often can combine with other morphemes too.
  house (*house-s*), *walk* (*walk-ed*), *of*, *the*, *or*

Past tense morpheme is a bound morpheme in English (*-ed*) but a free morpheme in Mandarine Chinese (*le*)

(3) a. *Ta chi le fan.*
He eat past meal.
'He ate the meal.'
b. *Ta chi fan le.*
He eat meal past.
'He ate the meal.'

Root × Affix

- **Root** – nucleus of the word that affixes attach too.
  In English, most of the roots are free. In some languages that is less common (Lithuanian: *Billas Clintonas*).
  Some words (compounds) contain more than one root: *home-work*
- **Affix** – a morpheme that is not a root; it is always bound
  - **suffix**: follows the root
    English: *-ful* in *event-ful*, *talk-ing*, *quick-ly*, *neighbor-hood*
    Russian: *-a* in *ruk-a* 'hand'

  - **prefix**: precedes the root

    English: *un-* in *unhappy*, *pre-existing*, *re-view*
    Classical Nahuatl: *no-cal* 'my house'

  - **infix**: occurs inside the root

    English: very rare: *abso-bloody-lutely*
    Khmer: *-b-* in *lbeun* 'speed' from *leun* 'fast';
    Tagalog: *-um-* in *s-um-ulat* 'write'

    The places in the stem where infixing can occur are quite restricted: either in the second or prefinal position, where various units are counted – syllables, moras, consonants, vowels, etc. Hoeksema and Janda:212 (11)

  - **circumfix**: occurs on both sides of the root

    Tuwali Ifugao *baddang* 'help', *ka-baddang-an* 'helpfulness', **ka-baddang*, **baddang-an*;
    Dutch: *berg* 'mountain' *ge-berg-te*, 'mountains' , **geberg*, **bergte vogel* 'bird', *ge-vogel-te* 'poultry', **gevogel*, **vogelte*

Suffixing is more frequent than prefixing and far more frequent than infixing/circumfixing (Greenberg 1957; Hawkins and Gilligan 1988; Sapir 1921). It is important that the asymmetry holds not only when simply counting languages, which is always problematic, but also in diverse statistical measures. Hawkins and Gilligan (1988) suggest a number of universals capturing the correlation between affix position in morphology and head position in syntax. The correlation is significantly skewed towards preference of suffixes. For example, postpositional and head-final languages use suffixes and no prefixes; while prepositional and head-initial languages use not only prefixes, as expected, but also suffixes. Moreover, there are many languages that use exclusively suffixes and not prefixes (e.g., Basque, Finnish), but there are very few that use only prefixes and no suffixes (e.g., Thai, but in derivation, not in inflection). There have been several attempts to explain the suffix-prefix asymmetry, using processing arguments (Cutler et al. 1985; Hawkins and Gilligan 1988), historical arguments (Givón 1979) , and combinations of both (Hall 1988) (see Hana and Culicover 2008 for an overview).

Content × Functional

- **Content** morphemes – carry some semantic content
  car, *-able*, *un-*

- **Functional** morphemes – provide grammatical information
  the, *and*, *-s* (plural), *-s* (3^{rd} sg)

Inflection × Derivation

There are two rather different kinds of morphological relationship among words, for which two technical terms are commonly used:

- **Inflection**: creates new forms of the same lexeme.

  E.g., *bring*, *brought*, *brings*, *bringing* are inflected forms of the lexeme BRING.

- **Derivation**: creates new lexemes

  E.g., *logic*, *logical*, *illogical*, *illogicality*, *logician*, etc. are derived from *logic*, but they all are different lexemes now.

- **Ending** – inflectional suffix

- **Stem** – word without its inflectional affixes = root plus all derivational affixes.

Derivation tends to affects the meaning of the word, while inflection tends to affect only its syntactic function. Also, derivation tends to be more irregular – there are more gaps, the meaning is more idiosyncratic and less compositional. However, the boundary between derivation and inflection is often fuzzy and unclear.

2.3 Morphological processes

- **Concatenation** (adding continuous affixes) – the most common process

  Often, there are phonological changes on morpheme boundaries.

- **Reduplication** – part of the word or the entire word is doubled:

  - Tagalog: *basa* 'read' – *ba-basa* 'will read'; *sulat* 'write' – *su-sulat* 'will write'
  - Afrikaans: *amper* 'nearly' – *amper-amper* 'very nearly'; *dik* 'thick' – *dik-dik* 'very thick'
  - Indonesian: *oraŋ* 'man' – *oraŋ-oraŋ* 'all sorts of men' (Cf. *orangutan*)
  - Samoan:

alofa	'love$_{Sg}$'	*a-lo-lofa*	'love$_{Pl}$'
galue	'work$_{Sg}$'	*ga-lu-lue*	'work$_{Pl}$'
la:poʔa	'to be large$_{Sg}$'	*la:-po-poʔa*	'to be large$_{Pl}$'
tamoʔe	'run$_{Sg}$'	*ta-mo-moʔe*	'run$_{Pl}$'

  - English: *humpty-dumpty*
  - American English (borrowed from Yiddish): *baby-schmaby, pizza-schmizza*

- **Templates** – both the roots and affixes are discontinuous. Only Semitic lgs (Arabic, Hebrew).

  Root (3 or 4 consonants, e.g., *l-m-d* – 'learn') is interleaved with a (mostly) vocalic pattern

  - Hebrew:

lomed	'learn$_{masc}$'	shotek	'be-quiet$_{pres.masc}$'
lamad	'learned$_{masc.sg.3rd}$'	shatak	'was-quiet$_{masc.sg.3rd}$'
limed	'taught$_{masc.sg.3rd}$'	shitek	'made-sb-to-be-quiet$_{masc.sg.3rd}$'
lumad	'was-taught$_{masc.sg.3rd}$'	shutak	'was-made-to-be-quiet$_{masc.sg.3rd}$'

- **Suppletion** – 'irregular' relation between the words. Hopefully quite rare.

- Czech: *být 'to be'* – *jsem 'am'*, *jít* 'to go' – *šla* 'went$_{fem.sg}$', *dobrý* 'good' – *lepší* 'better'
- English: *be* – *am* – *is* – *was*, *go* – *went*, *good* – *better*

- **Morpheme internal changes** (apophony, ablaut) – the word changes internally
  - English: *sing* – *sang* – *sung*, *man* – *men*, *goose* – *geese* (not productive anymore)
  - German: *Mann* 'man' – *Männ-chen* 'small man', *Hund* 'dog' – *Hünd-chen* 'small dog'
  - Czech: *kráva* 'cow$_{nom}$' – *krav* 'cows$_{gen}$', *nés-t* 'to carry' – *nes-u* 'I am carrying' – *nos-ím* 'I carry'
- **Subtraction (Deletion)**: some material is deleted to create another form
  - Papago (a native American language in Arizona) imperfective → perfective

him	'walking$_{imperf}$'	→	*hi*	'walking$_{perf}$'
hihim	'walking$_{pl.imperf}$'	→	*hihi*	'walking$_{pl.perf}$'

  - French, feminine adjective → masculine adj. (much less clear)

grande [grɑ̃d]	'big$_f$'	→	*grand* [grɑ̃]	'big$_m$'
fausse [fos]	'false$_f$'	→	*faux* [fo]	'false$_m$'

2.4 Word formation: some examples

- **Affixation** – words are formed by adding affixes.
  - V + *-able* → Adj: *predict-able*
  - V + *-er* → N: *sing-er*
  - *un* + A → A: *un-productive*
  - A + *-en* → V: *deep-en*, *thick-en*
- **Compounding** – words are formed by combining two or more words.
  - Adj + Adj → Adj: *bitter-sweet*
  - N + N → N: *rain-bow*
  - V + N → V: *pick-pocket*
  - P + V → V: *over-do*
- **Acronyms** – like abbreviations, but acts as a normal word
  laser – light amplification by simulated emission of radiation
  radar – radio detecting and ranging
- **Blending** – parts of two different words are combined
  - *breakfast + lunch → brunch*
  - *smoke + fog → smog*
  - *motor + hotel → motel*
- **Clipping** – longer words are shortened
  - *doctor, professional, laboratory, advertisement, dormitory, examination*
  - *bicycle* (bike)
  - *refrigerator*

2.5 Morphological types of languages

Morphology is not equally prominent in all languages. What one language expresses morphologically may be expressed by different means in another language. For example, to express the aspect, English uses certain syntactic structures, such as

(4) a. John wrote (AE)/ has written a letter. (the action is complete)
b. John was writing a letter (process).

Other languages, such as Russian, for example, use a prefix to express similar meaning, for example:

(5) a. John **na**pisal pis'mo. (the action is complete)
b. John pisal pis'mo. (process).

There are two basic morphological types of language structure:

- **Analytic** languages – have only free morphemes, sentences are sequences of single-morpheme words.

  (6) Vietnamese:

  khi tôi đên nhà bạn tôi, chúng tôi bat dăù làm bài
  when I come house friend I, PLURAL I begin do lesson

  When I came to my friend's house, we began to do lessons.

- **Synthetic** – both free and bound morphemes. Affixes are added to roots.

  Has further subtypes:

  – **Agglutinating** – each morpheme has a single function, it is easy to separate them.

  E.g., Uralic lgs (Estonian, Finnish, Hungarian), Turkish, Basque, Dravidian lgs (Tamil, Kannada, Telugu), Esperanto

  Turkish:

	singular	plural	
nom.	ev	ev-ler	'house'
gen.	ev-in	ev-ler-in	
dat.	ev-e	ev-ler-e	
acc.	ev-i	ev-ler-i	
loc.	ev-de	ev-ler-de	
ins.	ev-den	ev-ler-den	

  – **Fusional** – like agglutinating, but affixes tend to "fuse together", one affix has more than one function.

  E.g., Indo-European: Germanic (English, German, . . .), Romance languages (French, Spanish, . . .), Slavic (Russian, Czech, Polish, . . .), Greek, Semitic, Sami (Skolt Sami, . . .)

  For example,

  ∗ Czech *matk-a* 'mother' – *a* means the word is a noun, feminine, singular, nominative.

∗ In Serbian/Croatian nouns the number and case is expressed by one suffix:

	singular	plural	
nominative	ovc-a	ovc-e	‘*ovca* ‘sheep’
genitive	ovc-e	ovac-a	
dative	ovc-i	ovc-ama	
accusative	ovc-u	ovc-e	
vocative	ovc-o	ovc-e	
instrumental	ovc-om	ovc-ama	

Clearly, it is not possible to isolate separate singular or plural or nominative or accusative (etc.) morphemes.

– **Polysynthetic**: extremely complex, many roots and affixes combine together, often one word corresponds to a whole sentence in other languages.

angyaghllangyugtuq – ’he wants to acquire a big boat’ (Eskimo)
palyamunurringkutjamunurtu – ’s/he definitely did not become bad’ (W Aus.)
Sora

English has many analytic properties (future morpheme *will*, perfective morpheme *have*, etc. are separate words) and many synthetic properties (plural (*-s*), etc. are bound morphemes).

The distinction between analytic and (poly)synthetic languages is not a bipartition or a tripartition, but a continuum, ranging from the most radically isolating to the most highly polysynthetic languages. It is possible to determine the position of a language on this continuum by computing its degree of synthesis, i.e., the ratio of morphemes per word in a random text sample of the language. Table 2.5 gives the degree of synthesis for a small selection of languages. This table is taken from (Haspelmath 2002).

Language	Ration of morphemes per word
Greenlandic Eskimo	3.72
Sanskrit	2.59
Swahili	2.55
Old English	2.12
Lezgian	1.93
German	1.92
Modern English	1.68
Vietnamese	1.06

Table 2.1: The degree of synthesis of some languages

2.6 Some difficulties in morpheme analysis

Zero morpheme

Another phenomenon that causes problems for splitting words into morphemes, is the existence of forms in which a morphological meaning corresponds to no overt formal element; this is generally called **zero morpheme**. Here are several examples:

- Coptic:

ǰo-i	'my head'
ǰo-k	'your (masc.) head'
ǰo	'your (fem.) head'
ǰo-f	'his head'
ǰo-s	'her head'

- Finnish:

oli-n	'I was'
oli-t	'you were'
oli	'he/she was'
oli-mme	'we were'
oli-tte	'you (pl.) were'
oli-vat	'they were'

Some morphologists have worked with the requirement that the segmentation of words into morphemes must be exhaustive and all meanings must be assigned to a morpheme. If one adopts this requirement, then one is forced to posit zero morphemes here that have a meaning, but no form (for Finnish *oli* would really have the structure *oli-Ø*, where the morpheme Ø stands for the third person singular). But the requirement is not necessary, and alternatively one could say, for instance, that Finnish has no marker for the third person singular in verbs.

Empty morphemes

The opposite of zero morphemes can also be found: apparent cases of morphemes that have form but no meaning, called *empty morphemes.* For example, in Lezgian all nominal case-forms except for the absolutive case (i.e., the most basic case) contain a suffix that follows the noun stem and precedes the case suffix:

- Four of Lezgian's sixteen cases:

absolutive	sew	fil	Rahim
genitive	sew-re-n	fil-di-n	Rahim-a-n
dative	sew-re-z	fil-di-z	Rahim-a-z
subessive	sew-re-k	fil-di-k	Rahim-a-k
	'bear'	'elephant'	(male name)

This suffix, called the *oblique stem* suffix in Lezgian grammar, has no meaning, but it must be posited if we want to have an elegant description. With the notion of an empty morpheme we can say that different nouns select different suppletive oblique stem suffixes, but that the actual case suffixes that are affixed to the oblique stem are uniform for all nouns.

What is an alternative analysis?

Clitics

Clitics are units that are transitional between words and affixes, having some properties of words and some properties of affixes, for example:

- Unlike words:

  - Placement of clitics is more restricted.
  - Cannot stand in isolation.
  - Cannot bear contrastive stress.
  - etc.

- Unlike affixes, clitics:
  - Are less selective to which word (their host) they attach, e.g. host's part-of-speech may play no role.
  - Phonological processes that occur across morpheme boundary do not occur across host-clitic boundary.
  - etc.

The exact mix of these properties varies considerably across languages. The way clitics are spelled varies considerably not only across languages but also within a single language. Clitics are written as affixes of their host, sometimes are separated by punctuation (e.g., possessive *'s* in English) and sometimes are written as separate words. For more details see (Anderson 1993; Franks and King 2000; Hana 2007; Klavans 1982; Zwicky 1977) and references therein.

2.7 Example: Rich Inflection – Czech and Russian

Table 2.2 shows an example of two parallel non paradigms from Czech and Russian. In both languages, nominal categories (adjectives, nouns, pronouns) inflect for gender, number, case. Both languages have 3 genders (masculine, feminine, neuter) and two numbers (Czech has also some remnants of dual number). They share 6 cases with roughly the same meaning (nominative, genitive, dative, accusative, local, instrumental). In addition, Czech has vocative and Russian has two secondary cases: second genitive and second locative. In both languages, nouns are grouped into declension classes. Numerals use declensional strategies which range from near indeclinability to adjective-like declension. Neither language has articles; (in)definiteness is expressed using other means, e.g., word order.

Morphology in both languages exhibits

- a high number of fusion – several morphemic categories whose values are combined in clusters, each of which is expressed by a single ending (e.g., number, gender, and case with nouns or adjectives, or tense, number, and person with finite verbs), and
- a high degree of ambiguity of the endings. See Tables 2.3 and 2.4 for examples.[1].
- a relatively common synonymy of the endings.

Simply put, in a fusional language like Russian or Czech, a paradigm is a set of endings with their tags, e.g., *0* – noun singular, *s* – noun plural. The endings are added to stems producing word forms characterized by those tags, e.g., *cat* – noun singular, *cats* – noun plural. However, life is not easy, and the concatenation is often accompanied by various more or less complicated phonological/graphemic processes affecting the stem, the ending or both, e.g., *potato-**es**, countr**i**-es, kni**v**-es*, etc.

[1]Abbreviations of morphological categories (e.g., `FS1` – feminine singular nominative, `P4` – plural accusative) are based on Hajič (2004) tagset (see §5.1)

	Czech	Russian	Gloss
sg.			
nom	žen-a	ženščin-a	‘woman’
gen	žen-y	ženščin-y	
dat	žen-ě	ženščin-e	
acc	žen-u	ženščin-u	
voc	žen-o	–	
loc	žen-ě	ženščin-e	
ins	žen-ou	ženščin-oj/ou	
pl.			
nom	žen-y	ženščin-y	
gen	žen	ženščin	
dat	žen-ám	ženščin-am	
acc	žen-y	ženščin	
voc	žen-y	–	
loc	žen-ách	ženščin-ax	
ins	žen-ami	ženščin-ami	

Table 2.2: Czech and Russian noun declension

Table 2.3: Homonymy of the *a* ending in Czech

form	lemma	gloss		category
měst-a	město	town	NS2	noun neut sg gen
			NP1 (5)	noun neut pl nom (voc)
			NP4	noun neut pl acc
tém-a	téma	theme	NS1 (5)	noun neut sg nom (voc)
			NS4	noun neut sg acc
žen-a	žena	woman	FS1	noun fem sg nom
pán-a	pán	man	MS2	noun masc anim sg gen
			MS4	noun masc anim sg acc
ostrov-a	ostrov	island	IS2	noun masc inanim sg gen
předsed-a	předseda	president	MS1	noun masc anim sg nom
vidě-l-a	vidět	see		verb past fem sg
				verb past neut pl
vidě-n-a				verb passive fem sg
				verb passive neut pl
vid-a				verb transgressive masc sg
dv-a	dv-a	two		numeral masc sg nom
				numeral masc sg acc

Table 2.4: Ending *-e* and noun cases in Czech

case	form	lemma	gender	gloss
nom	kuř-e	kuře	neuter	chicken
gen	muž-e	muž	masc.anim.	man
dat	mouš-e	moucha	feminine	fly
acc	muž-e	muž	masc.anim.	man
voc	pan-e	pán	masc.anim.	mister
loc	mouš-e	moucha	feminine	fly
inst	–	–		

Agreement in Czech and Russian Adjectives and possessives agree in gender, number and case with the noun they modify. In Russian, gender agreement is only in singular. Main verbs agree in person and number with their subjects. Past participles agree with subject in number and gender (again, in Russian gender agreement is only in singular).

(7) a. [Czech]

Byl	jasný,	studený	dubnový	den	a
was$_{masc.past}$	bright$_{masc.sg.nom}$	cold$_{masc.sg.nom}$	April$_{masc.sg.nom}$	day$_{masc.sg.nom}$	and

hodiny	odbíjely	třináctou.
clocks$_{fem.pl.nom}$	stroke$_{fem.pl.past}$	thirteenth$_{fem.sg.acc}$

b. [Russian]

Byl	jasnyj,	xolodnyj	aprel'skij	den'	i
was$_{masc.past}$	bright$_{masc.sg.nom}$	cold$_{masc.sg.nom}$	April$_{masc.sg.nom}$	day$_{masc.sg.nom}$	and

časy	probili	trinadtsat'.
clocks$_{pl.nom}$	stroke$_{pl.past}$	thirteen$_{acc}$

'It was a bright cold day in April, and the clocks were striking thirteen.' [from Orwell's '1984']

Word order Both Czech and Russian are free-word-order languages. Syntactic relations within a sentence are expressed by inflection and the order of constituents is determined mainly by pragmatic constraints. The theme (roughly old information) usually precedes the rheme (roughly new information). There are, however, certain rigid word order combinations, such as noun modifiers, clitics (in Czech), and negation (in Russian).

Morphologically rich language always have a relatively free word order because the grammatical functions can be expressed in morphology rather than by a position in a sentence.

Czech noun paradigms

Table 2.5: Examples of the *žena* paradigm nouns

	woman	owl	draft	goat	iceberg	vapor	fly	
S1	žen-a	sov-a	skic-a	koz-a	kr-a	pár-a	mouch-a	
S2	žen-y	sov-y	skic-**i**	koz-y	kr-y	pár-y	mouch-y	
S3	žen-ě	sov-ě	skic-**e**	koz-**e**	k**ř-e**	pá**ř**-e	mou**š-e**	
S4	žen-u	sov-u	skic-u	koz-u	kr-u	pár-u	mouch-u	
S5	žen-o	sov-o	skic-o	koz-o	kr-o	pár-o	mouch-o	
S6	žen-ě	sov-ě	skic-**e**	koz-**e**	k**ř-e**	pá**ř**-e	mou**š-e**	
S7	žen-ou	sov-ou	skic-ou	koz-ou	kr-ou	pár-ou	mouch-ou	
P1	žen-y	sov-y	skic-**i**	koz-y	kr-y	pár-y	mouch-y	
P2	žen-0	sov-0	skic-0	koz-0	k**e**r-0	p**a**r-0	m**u**ch-0	
P3	žen-ám	sov-ám	skic-ám	koz-ám	kr-ám	pár-ám	mouch-ám	
P4	žen-y	sov-y	skic-**i**	koz-y	kr-y	pár-y	mouch-y	
P5	žen-y	sov-y	skic-**i**	koz-y	kr-y	pár-y	mouch-y	
P6	žen-ách	sov-ách	skic-ách	koz-ách	kr-ách	pár-ách	mouch-ách	
P7	žen-ami	sov-ami	skic-ami	koz-ami	kr-ami	pár-ami	mouch-ami	

As a more complex illustration, consider several examples of Czech nouns belonging to the *žena* 'woman' paradigm, a relatively 'well-behaved' paradigm of feminine nouns, in Table 2.5. Without going too deeply into linguistics, we can see several complications:

1. Ending variation: *žen-ě*, *sov-ě* vs. *burz-e*, *kř-e*, *pář-e*; *žen-y* vs. *skic-i*.

   a) The dative and local sg. ending is *-ě* after alveolar stops (*d, t, n*) and labials (*b, p, m, v, f*). It is *-e* otherwise.

   b) Czech spelling rules require the ending *-y* to be spelled as *-i* after certain consonants, in this case: *c, č, ď, ň, š*. The pronunciation is the same ([ɪ]).

2. Palatalization of the stem final consonant: *kr-a – kř-e*, *mouch-a – mouš-e*.

   The *-ě/e* ending affects the preceding consonant: *ch* [x] → *š*, *g/h* → *z*, *k* → *c*, *r* → *ř*.

3. Epenthesis: *kr-a – ker*.

   Sometimes, there is an epenthesis (insertion of *-e-*) in genitive plural. This usually happens when the noun ends with particular consonants. There are certain tendencies, but in the end it is just a property of the lexeme; cf. *občank-a – občanek* 'she-citizen, id-card' vs. *bank-a – bank* 'bank' (both end with *nk*, but one epenthesises and the other not). Some nouns allow both possibilities, e.g., *jacht-a – jachet/jacht* 'yacht'

4. Stem internal vowel shortening: *pár-a – par*.

   Often the vowels *á, í, ou* shorten into *a, i/ě, u* in gen. pl. and sometimes also in dat., loc. and ins. pl. If the vowel is followed by multiple consonants in nom. sg, the shortening usually does not happen. In many cases there are both short and long variants (*pár-a – par – pár-ám/par-ám*, *pár-ách/par-ách*, *pár-ami/par-ami* 'vapor'), usually stylistically different.

It would be possible to discuss in a similar manner all the Czech noun paradigms. Depending on how you count, there are roughly 13 basic paradigms – 4 neuter, 3 feminine and 6 masculine; plus there are nouns with adjectival declension (another 2 paradigms). In addition, there are many subparadigms and subsubparadigms, all of which involves a great amount of irregularity and variation on the one hand and a great amount of homonymy on the other (see Table 2.3). Also, some forms have official and colloquial variants. For a more detailed discussion, see for example (Fronek 1999; Karlík et al. 1996).

Also note that in Czech, there is a significant difference in morphology and the lexicon between the standard and colloquial levels of Czech. The automatic morphological analysis of such a language is especially challenging since the same word can have several morphological forms, depending on the language level. It also means that a tagset of Czech (assuming it captures this feature) is significantly larger than tagset of another Slavic language, with otherwise comparable morphology.

Chapter 3

Classical Approaches to Computational Morphological Analysis

The discussion in this section is largely based on (Roark and Sproat 2007) and (Goldsmith 2010).

3.1 What is morphological analysis?

Suppose you are studying Russian. You come across an unfamiliar form: *pišu* and you want to know what it means. You already know the verb *pisal* 'write.3P.sg.imperf.active.indicative' and you guess that the unfamiliar form should be related to the one you already know.

- *pišu*: 'write', 1ST PERSON, SINGULAR, IMPERFECT, ACTIVE, PRESENT, INDICATIVE
- *pišu*: *piš-u* (*s* becomes *š*) → DECOMPOSITION
- *pišu*: lemma: *pisat'* → LEMMATIZATION = the task of relating a given form to a canonical form.

Applications What are the applications of morphological analysis?

- parsing/chunking (used in machine translation, grammar correction, etc.)
- Search and information retrieval. One usually searches for a lexeme not for a particular form.
- For applications, such as text-to-speech synthesis, we may be also interested in the analysis of a word into its parts.

  For example, in order to determine which syllable of a given instance of the Russian word *snega* should be stressed, one must know the morphological properties of that instance — the genitive singular form of the word is stressed on the first syllable, while the nominative plural form is stressed on the second:

  snèga.Noun.*Gen.Masc.Singular* 'snow' vs. *snegà*.Noun.*Nom-Acc.Plural* 'snows'.
- spell checking

Formally, lemmatization can be viewed as a combination of decomposition followed by normalization to the lemmatized form.

Complications Morphological analysis is difficult. Some potential problems include:

- Stem internal (non-concatenative) alternations: e.g., the German *Stuhl* → *Stühle*, *Vater* → *Väter.*
- Irregularities: e.g., the Russian plural forms, e.g., *kniga*→ *knigi*, *stol* → *stoly* , but *kofe* → *kofe*; or the English, *goose* → *geese*, *sheep* → *sheep.*
- Phonological/graphemic alternations
- Homonymy: e.g., English *-s* marks both 3rd person singular of verbs and plural of nouns; Czech *-a* (see Table 2.3).

Thus, morphological analysis can be viewed as a function that assigns a set of lemmas (base forms), each with a set of tags, to a form:

(8) MA: form → set(lemma × set(tag))

ženou	→	{ (*žena* 'woman',	{noun fem sing inst }),
		(*hnát* 'hurry',	{verb pres pl 3rd }) }
ženy	→	{ (*žena* 'woman',	{noun fem sing gen,
			noun fem pl nom,
			noun fem pl acc,
			noun fem pl voc }) }

Note: In this chapter, we focus on inflection expressed by suffixes, leaving other types of morphological systems (templates, reduplication, etc) aside.

3.2 Different Approaches

There are two different ways to address phonological/graphemic variations and complex paradigm systems when designing a morphological analyzer:

- A linguistic approach. Such a system employs a phonological component accompanying the simple concatenative process of attaching an ending. This implies a smaller set of paradigms and morphemes. Two-level morphology (Koskenniemi 1983a, 1984) is an example of such a system and (Skoumalová 1997) is an example for Czech. The problem is that implementing morphology of a language in such a system requires a lot of linguistic work and expertise. For many languages, the linguistic knowledge is not precise enough. Moreover, it is usually not straightforward to translate even a precisely formulated linguistic description of a morphology into the representation recognized by such system.

  In Czech, the forms of the noun *kra* 'iceberg$_{FS1}$', *kře* 'iceberg$_{FS36}$', *ker* 'iceberg$_{FP2}$' etc. (see Table 2.5) would be analyzed as involving the stem *kr*, the endings *-a*, *-ě* and *-0* and phonological/graphemic alternations. Forms of the noun *žena* 'woman$_{FS1}$' (*ženě* 'FS36', *žen* 'FP2', etc.) would belong to the same paradigm as *kra.*

- An engineering approach. Such a system does not have a phonological component, or the component is very rudimentary. Phonological changes and irregularities are factored into endings and a higher number of paradigms. This implies that the terms *stem* and *ending* have slightly different meanings than they traditionally do. A stem is the part of the word that does not change within its paradigm, and the ending is the part of the word that follows such a stem.

  Examples of such an approach are (Hajič 2004) for Czech and (Mikheev and Liubushkina 1995) for Russian. The previous version of our system (Hana et al. 2004) also belongs to this category. The advantages of such a system are its high speed, simple implementation and straightforward morphology specification. The problems are a very high number of paradigms (several hundreds in the case of Czech) and impossibility to capture even the simplest and most regular phonological changes and so predict the behavior of new lexemes.

  For example, the English noun paradigm above (*0 – s*) would be captured as several other paradigms including, *0 – s*, *0 – es*, *y – ies*, *f – ves*.

  In Czech, the forms of the noun *kra* 'iceberg$_{FS1}$' would be analyzed as involving the stem *k* followed by the endings *-ra*, *-ře* and *-er*. Forms of the nouns *žena* 'woman$_{FS1}$' and *kra* would belong to two different paradigms.

3.3 Linguistic Approach: Finite-State Morphology

Finite-state morphology aims at analyzing morphology within the computational power of finite-state automata. It is by far the most popular approach in the field. Finite-state approaches to morphology provide ways of analyzing surface forms by appealing to the notion of a finite-state transducer which in turn mimics an ordered set of rewrite rules. The finite state approaches to morphology were stimulate by the work of (Johnson 1972; Kaplan and Kay 1981; Koskenniemi 1983b) and was extensively discussed in (Beesley and Karttunen 2003).

What is finite state technology?

The notion of finite state automaton (often abbreviated as FSA) was first presented in (Kleene 1956). An FSA is a kind of directed graph: a directed graph is by definition a finite set of nodes N, along with a set of edges E, where an edge is an ordered pair of nodes. Nodes in an FSA are often called states. For a directed graph to be an FSA, it must be endowed with three additional properties:

- It must have a distinguished node identified as its start state;
- it must have a set of one or more stopping (or accepting) states;
- and it must have a set of labels, L, with each edge associated with exactly one label in L.

While L cannot in general be null, it may contain the null string as one of its members.

For example, the morphs of a given language will be members of L, as will be descriptions of grammatical feature specifications, such as 1st person or past-tense. We are typically interested in the set of paths through the graph, and the strings associated with each such path.

A path in a given FSA is defined as a sequence of nodes selected from N, in which the first node in the sequence is the starting state of the FSA, the last node in the sequence is one of the stopping states of the FSA, and each pair of successive nodes (n_i, n_{i+1}) in the sequence corresponds to an edge e_j of the FSA. We associate a string S with a path p simply by concatenating all of the labels of the edges corresponding to the successive pairs of nodes comprising p.

If we take a grammar of a language to be a formal device which identifies a set of grammatical strings of symbols, then an FSA is a grammar, because it can be used to identify the set of strings that correspond to all paths through it. Given a string S in L, we can identify all paths through the FSA that generate S.

Finite state transducers (FSTs)

Finite state morphologies employ a generalization of the finite-state automaton called a finite state transducer, or FST, following work by (Johnson 1972). An FST differs from an FSA in that an FST has two sets of labels (or in principle even more), one called underlying labels, LU, and one called surface labels, LS, and each edge is associated with a pair of labels (lU, lS), the first chosen from the underlying labels, and the second from the surface labels. In fact, an FST can be thought of as two (or even more) FSAs which share the same nodes, edges, starting states and stopping states, but which differ with regard to the labels associated with each edge, and we only care about looking at pairs of identical paths through these two FSAs. FST allows us to think about pairs of parallel paths through otherwise identical FSAs as if they were just a single path through a single directed graph.

FSTs and morphology

A large part of the work on computational morphology has involved the use of finite-state devices, including the development of computational tools and infrastructure. Finite state methods have been used to handle both the strictly morphological and morphotactic, on the one hand, and the morphophonology and graphotactics on the other. By extending the notion of finite state automaton to that of finite state transducer, it is possible to not only generate the correct surface morphemes, but also create a device that can map surface sequences of letters (or phones) to abstract morphosyntactic features such as number and tense.

Computational morphology has also applied the notion of finite state transducer to deal with the problem of accounting for regularities of various sorts concerning alternative ways of realizing morphemes. For example, both the English nominal suffix marking plural and the English verbal suffix marking 3rd person singular is normally realized as *s*, but both are regularly realized as *es* after a range of stems which end in *s, sh, ch*, and *z*. These aspects are called morphotactics and phonology, respectively. Two methods have been developed in considerable detail for the implementation of these two aspects within the context of finite state devices. One, often called *two level morphology*, is based on an architecture in which a set of constraints are expressed as finite-state transducers that apply in parallel to an underlying and a surface representation. Informally speaking, each such transducer acts like a constraint on possible differences that are permitted between the underlying and the surface labels, and as such, any paired underlying/surface string must satisfy all transducers. The other approach involves not a parallel set of finite-state transducers, but rather a cascaded set of finite-state transducers, which can be compiled into a single transducer. A history of this work, with an account of the relationship between these approaches, can be found in (Karttunen and Beesley Saarijärvi, Finland) and in (Karttunen 1993).

Two-level morphology

Two level morphology was first proposed by (Koskenniemi 1983a). Two-level morphology represents a word as a correspondence between a **lexical level**, which represents a simple concatenation of morphemes making up a word, and the **surface level**, which represents the actual spelling of the final word. Morphological analysis is implemented by building mapping rules that map letter sequences like *cats* on the surface level into morpheme and features sequences like *cat +N +PL* on the lexical level. Here is an example. Imagine that the formation of the plural form of *shelf* could be broken up into successive stages: *shelf*, *shelf +s*, *shelv +s*, *shelves*. Here, we see the suffixation of the plural *es* happening first, followed by the change of *f* to *v*, followed in turn by the insertion of *e*. In contrast, finite-state automata offer a way of dealing with the central phenomena of morphology without recourse to such a step-by-step derivation: hence the term *two-level morphology*, which employs only two levels: one in which morphosyntactic features and lexical roots are specified, and one which matches the spelled (or pronounced) form of the word.

(Koskenniemi 1983a)'s approach does not depend on a rule compiler, composition or any other finite-state algorithm. Rules can be thought of as statements that directly constrain the surface realization of lexical strings. The rules are applied in parallel.

Two-level morphology is based on three ideas:

1. Rules are symbol-to-symbol constraints that are applied in parallel, not sequentially like rewrite rules.
2. The constraints can refer to the lexical context, to the surface context, or to both contexts at the same time.
3. Lexical lookup and morphological analysis are performed in tandem.

We will provide a more detailed example in class.

3.4 Engineering Approach

An example of the engineering approach is the work by Mikheev and Liubushkina. (Mikheev and Liubushkina 1995)'s main objective is an efficient computational recognition of morpho-syntactic features of words and the generation of words according to requested morpho-syntactic features. The authors develop a template-oriented word paradigm model. Paradigm formation rules are bottom-up (word specific) and word formation units are segments of words rather than proper morphemes. Such an approach has many advantages: the approach handles uniformly both general cases and exceptions and requires simple data structures and control mechanisms which can be implemented as a finite state automaton. The morphological processor is fully implemented for a 1,500,000 word token (38,000 stems) corpus and provides morpho-syntactic features and stress positions. Special dictionary management tools are built for browsing, debugging and extension of the lexicon. The first 5,000 paradigms entered explicitly into the system provide about 400 paradigmatic classes[1] which are able to cover more than 80% of the lexicon. Though Mikheev and Liubushkina 1995's approach does not use extensive resources, the system still relies on a dictionary of frequent words with 4,000 stems which correspond to 9,500 lexemes (about 120,000 word tokens).

[1]Here a paradigmatic class corresponds to the concept of a paradigm used in our work, i.e. a set of endings which is the realization of all paradigmatic variations for the lexeme.

Chapter 4

Classical tagging techniques

4.1 What is morphological tagging?

Part-of-speech (POS) tagging is the task of labeling each word in a sentence with its appropriate POS information. Morphological tagging is very similar. It is a process of labeling words in a text with their appropriate detailed morphological information.

4.2 Supervised vs. Unsupervised tagging

There are many approaches to automated POS tagging. One of the first distinctions which can be made among POS taggers is in terms of the degree of automation of the training and tagging process. The terms commonly applied to this distinction are *supervised* vs. *unsupervised*.

- *Supervised* taggers typically rely on pretagged corpora to serve as the basis for creating any tools to be used throughout the tagging process, such as the tagger dictionary, the word/tag frequencies, the tag sequence probabilities and/or the rule set.
- *Unsupervised* models, on the other hand, are those which do not require a pretagged corpus but instead use sophisticated computational methods to automatically induce word groupings (i.e., tagsets) and, based on those automatic groupings, to either calculate the probabilistic information needed by stochastic taggers or to induce the context rules needed by rule-based systems.

Each of these approaches has its pros and cons.

- It is known that supervised POS taggers tend to perform best when both trained and used on the same genre of text. The unfortunate reality is that pretagged corpora are not readily available for the many language and genres which one might wish to tag.
- Unsupervised tagging addresses the need to tag previously untagged genres and languages in light of the fact that hand tagging of training data is a costly and time-consuming process. There are, however, drawbacks to fully unsupervised POS tagging. The word clusterings (i.e., automatically derived tagsets) which tend to result from these methods are very coarse, i.e., one loses the fine distinctions found in the carefully designed tag sets used in the supervised methods.

In this chapter, we focus on supervised taggers. Unsupervised taggers are discussed in §6.

All of the taggers discussed here use the surrounding local context (typically, a window of two or three words and/or tags) to determine the proper tag for a given corpus position.

4.3 Measures of success

The following measures are typically used for evaluating the performance of a tagger:

(9) $\text{Precision} = \frac{\text{Correctly-Tagged-Tokens}}{\text{Tokens-generated}}$

Precision measures the percentage of system-provided tags that were correct.

$\text{Recall} = \frac{\text{Correctly-Tagged-Tokens}}{\text{Tokens-in-data}}$

Recall measures the percentage of tags actually present in the input that were correctly identified by the system.

$\text{F-measure} = 2 * \frac{\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$

The F-measure (Rijsbergen 1979) provides a way to combine these two measures into a single metric.

4.4 N-gram taggers/Markov models

N-gram taggers (Brants 2000; Church 1988; DeRose 1988; Weischedel et al. 1993) limit the class of models considered to $n-1$th order Markov models. Recall that a Markov model (MM) is a doubly stochastic process defined over a set of *hidden states* $\{s_i \in S\}$ and a set of *output symbols* $\{w_j \in W\}$. There are two sets of probabilities involved.

- *Transition probabilities* control the movement from state to state. They have the form $P(s_k|s_{k-1} \dots s_{k-n+1})$, which encodes the assumption that only the previous n states are relevant to the current prediction.
- *Emission probabilities* control the emission of output symbols from the hidden states. They have the form $P(w_k|s_k)$, encoding the fact that only the identity of the current state feeds into the decision about what to emit.

In an HMM-based part-of-speech tagger, the hidden states are identified with part-of-speech labels, while the output symbols are identified either with individual words or with equivalence classes over these words (the latter option is taken by, for example (Cutting et al. 1992), because of the desire to reduce the data sparsity problem).

Taken together with a distribution over the initial state s_0 , the emission and transition probabilities provide a kth order Markov model of the tagging process.

$$P(s_0 \dots s_k, w_0 \dots w_k) = P(s_0) \prod^{k} {}_{i=0} P(w_i|s_i) P(s_{i+1}|s_i \dots s_{i-k+1})$$

This defines the joint probability of a tag sequence $s_0 \dots s_k$ and a word sequence $w_0 \dots w_k$.

For actual tagging, one must find the best possible path through the Markov model of states and transitions, based on the transition and emission probabilities. However, in practice, this is extremely costly, as multiple ambiguous words mean that there will be a rapid growth in the number of transitions between states. To overcome this, the Viterbi algorithm (Viterbi 1967) is commonly used. The main observation made by the Viterbi algorithm is that for any state, there is only one most likely path to that state. Therefore, if several paths converge at a particular state, instead of recalculating them all when calculating the transitions from this state to the next, less likely paths can be discarded, and only the most likely ones are used for calculations. So, instead of calculating the costs for all paths, at each state only the k-best paths are kept.

The terms Visible Markov model (VMM) and Hidden Markov models (HMM) are sometimes confused. In the case of the supervised training the formalism is really a mixed formalism. In training a VMM is constructed, but then it is treated as an HMM when it is put to use for tagging new corpora.

One major problem with standard n-gram models is that they must be trained from some corpus, and because any particular training corpus is finite, some perfectly acceptable n-grams are bound to be missing from it. That means that the n-gram matrix is sparse; it is bound to have a very large number of cases of putative zero-probability n-grams that should really have some non-zero probability. In addition, this maximum-likelihood estimation method produces poor estimates when the counts are non-zero but still small. The n-grams cannot use long-distance context. Thus, they always tend to underestimate the probability of strings that happen not to have occurred nearby in the training corpus. There are some techniques that can be used to assign a non-zero probability to unseen possibilities. Such procedures are called "smoothing" (e.g., (Chen and Goodman 1996)).

4.5 TnT (Brants 2000)

Trigrams'n'Tags (TnT) is a statistical Markov model tagging approach, developed by (Brants 2000). Contrary to the claims found in the literature about Markov model POS tagging, TnT performs as well as other current approaches, such as Maximum Entropy (see §4.8). One comparison has even shown that TnT performs significantly better than the Maximum Entropy model for the tested corpora (see (Brants 2000)). This section describes this tagger in more detail, since the experiments that are discussed in the subsequent chapter use this particular classifier.

The tagger is based on a trigram Markov model. The states of the model represent tags, outputs represent the words. Transition probabilities depend on the states, and thus on pairs of tags. Output (emission) probabilities only depend on the most recent category.

So, explicitly, for a given sequence of words $w_1, ...w_T$ of length T, the following is calculated:

$$(10)\ argmax_{t_1,...t_T}[\overset{T}{\prod}_{i=1} P(t_i|t_{i-1}, t_{i-2})P(w_i|t_i)]P(t_{T+1}|t_T)$$

$t_1, ...t_T$ are elements of the tagset, the additional tags t_{-1}, t_0, and t_{T+1} are beginning of sequence and end of sequence markers. As Brants mentions, using these additional tags, even if they stem from rudimentary processing of punctuation marks, slightly improves tagging results. This is different from formulas presented in other publications, which just stop with a "loose end" at the last word. If sentence boundaries are not marked in the input, TnT adds these tags if it encounters one of [.!?;] as a token.

Transitions and output probabilities are estimated from a tagged corpus, using maximum likelihood probabilities, derived from the relative frequencies.

4.6 Handling sparsity

As has been described above, trigram probabilities generated from a corpus usually cannot directly be used because of the sparsity problem. This means that there are not enough instances for each trigram to reliably estimate the probability. Setting a probability to zero because the corresponding trigram never occurred in the corpus is undesired, since it causes the probability of a complete sequence to be set to zero, making it impossible to rank different sequences containing a zero probability. The smoothing paradigm that brings the best results in TnT is linear interpolation of unigrams, bigrams, and trigrams. A trigram probability is estimated this way:

(11) $P(t_3|t_1,t_2) = \lambda_1 \hat{P}(t_3) + \lambda_2 \hat{P}(t_3|t_2) + \lambda_3 \hat{P}(t_3|t_1,t_2)$

$\hat{P}$ are maximum likelihood estimates of the probabilities, and $\lambda_1 + \lambda_2 + \lambda_3 = 1$, so P again represents probability distributions.

(Brants 2000) uses the context-independent variant of linear interpolation, where the values of the λs do not depend on the particular trigram; that yields better results than the context-dependent variant. The values of the λs are estimated by deleted interpolation. This technique successfully removes each trigram from the training corpus and estimates best values for the λs from all other n-grams in the corpus. Given the frequency counts for unigrams, bigrams, and trigrams, the weights can be very efficiently determined with a processing time that is linear in the number of different trigrams.

Special features

- Suffix analysis for handling unknown words

  To handle unknown words (Brants 2000) uses Samuelsson's 1993 suffix analysis, which seems to work best for inflected languages. Tag probabilities are set according to the word's ending. Suffixes are strong predictors for word classes (e.g., 98% of the words in the Penn Treebank corpus ending in *-able* are adjectives and the rest are nouns).

  The probability distribution for a particular suffix is generated from all words in the training set that share the same suffix of some predefined maximum length. The term suffix, as used in TnT (as well as in the work described in this book), means 'final sequence of characters of a word' which is not necessarily a linguistically meaningful suffix.

- Capitalization

  Additional information which is used in TnT is capitalization. Tags are usually not informative about capitalization, but probability distributions of tags around capitalized words are different from those not capitalized. The effect is large for languages such as English or Russian, and smaller for German, which capitalizes all nouns. (Brants 2000) uses flag c_i that is true if w_i is a capitalized word and false otherwise. These flags are added to the contextual probability distributions. Instead of $P(t_3|t_1,t_2)$, (Brants 2000) uses $P(t_3,c_3|t_1,c_1,t_2,c_2)$. This is equivalent to doubling the size of the tagset and using different tags depending on the capitalization.

- Reducing the processing time

  The processing time of the Viterbi algorithm is reduced by introducing a beam search. Each state that receives a δ value smaller than the largest δ divided by some threshold value θ is excluded from further processing. While the Viterbi algorithm is guaranteed to find the sequence of states with the highest probability, this is no longer true when beam search is added. However, as (Brants 2000) reports, for practical purposes and the right choice of θ, there is virtually no difference between the algorithm with and without a beam.

4.7 Transformation-based error-driven learning (TBL)

Transformation-based error-driven learning (TBL) (Brill 1995) is a technique which attempts to automatically derive rules for classification from the training corpus. The advantage over statistically-based tagging is that the rules are more linguistic and, thus, more easily interpretable. The supervised TBL employs not only a small, annotated corpus but also a large unannotated corpus. A set of allowable lexical and contextual transformations is predetermined by templates operating on word forms and word tokens, respectively. A general lexical/contextual template has the form: *"for a given word, change tag A to tag B if precondition C is true"*. An example of a specific rule from instantiated template, cited in (Brill 1995), is *"change the tagging of a word from **noun** to **verb** if the previous word is tagged as a **modal**"*. The set of allowable transformations used in (Brill 1995) permits tags to be changed depending on the previous (following) three tags and on the previous (following) two word forms, but other conditions, including wider contexts, could equally well be specified.

There are three main steps in the TBL training process:

1. From the annotated corpus, a lexicon is built specifying the most likely tag for a given word. Unknown words are tagged with the most frequently occurring tag in the annotated corpus.

2. Lexical transformations are learned to guess the most likely tag for the unknown words (i.e., words not covered by the lexicon).

3. Contextual transformations are learned to improve tagging accuracy.

The learning procedure is carried out over several iterations. During each iteration, the result of each transformation (i.e., an instantiation of a template) is compared to the truth and the transformation that causes the greatest error reduction is chosen. If there is no such transformation or if the error reduction is smaller than a specified threshold, the learning process is halted. The complexity of learning the cues is $O(L * N_{train} * R)$, where L is the number of prespecified templates, N_{train} is the size in words of training data and R is the number of possible template instances. The complexity of the tagging of test data is $O(T * N_{test})$, where T is the number of transformations and N_{test} is the test data size. This rule-based tagger trained on 600K of English text has a tagging accuracy of 96.9%.

Current approaches to TBL rely crucially on preselecting all and only the relevant templates for transformations. Failure to satisfy this condition will result in overtraining or under-performance.

4.8 Maximum Entropy

A third supervised learning approach is the Maximum Entropy (MaxEnt) tagger (Ratnaparkhi 1996), which uses a probabilistic model basically defined as

(12) $p(h,t) = \pi\mu \prod^{k}_{j=1} \alpha^{f_j(h,t)}{}_j,$

where h is a context from the set of possible words and tag contexts (i.e., so-called "histories"), t is a tag from the set of possible tags, π is a normalization constant, $\{\mu, \alpha_1, \alpha_2, ..., \alpha_k\}$ are the positive model parameters and $\{f_1, f_2, ..., f_k\}$ is a set of yes/no features (i.e., $f_i(h,t) \in \{0,1\}$).

Each parameter α_i (the so-called *feature-weight*) corresponds to the exactly one feature f_i, and features operate over the events (context, tag). For a current word, the set of specific contexts is limited to the current word, the preceding two words together with their tags, and the following two words. The positive model parameters are chosen to maximize the likelihood of the training data. An f_i is true (or equals 1) if a particular linguistic condition is met.

Features which are determined to be important to the task are constrained to have the same expected value in the model as in the training data. That is, consistency with the training data is maintained by asserting this equality holds, as shown in (13), where Ef_j is the expected value of f in the model and $\tilde{E}f_j$ is the empirical expected value of f in the training sample.

(13) $Ef_j = \tilde{E}f_j$

The features used in (Ratnaparkhi 1996) are derived from templates, similar to those in Brill 1995. For example, three templates are shown in (14), where w_i is the i'th word, t_i is the i'th tag, and X and T refer to values to be filled in.

(14)
1. X is a suffix of w_i, $|X| \leq 4$ & $t_i = T$
2. $t_{i-1} = X$ & $t_i = T$
3. $w_{i+1} = X$ & $t_i = T$

A feature f will be equal to one when the condition is met and zero otherwise. A feature has access to any word or tag in the history (h) of a given tag, as shown in (15).

(15) $h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$

So, for example, a feature might be as in (16).

(16) $f_j(h_i, t_i) = \left\{ \begin{array}{ll} 1 & \text{if suffix}(w_i) = \text{"ing" \& } t_i = \text{VBG} \\ 0 & \text{otherwise} \end{array} \right\}$

To set the features, the model will go through the training corpus asking yes/no questions about each item in h for a given tag t. From this, a tag obtains a given probability of being correct, based on its history.

When tagging a text, the joint probability of a tag t and its history h, i.e. $p(h,t)$, should be found. The joint probability is partly determined by the so-called *active* features, those features which have a value of one. The way the features determine the joint probability is by the constraint mentioned earlier, where the expected value for a feature f in the model must be equal to the empirical expected value for the feature. And the expected values are sums over the joint probabilities, as shown in (17), where H is the set of possible histories (word and tag contexts) and T is the set of possible tags. Thus, because $p(h,t)$ and $f_j(h,t)$ are involved in calculating Ef, the value of $p(h,t)$ is constrained by the value of $f_j(h,t)$.

(17) 1. $Ef_j = \sum_{h \in H, t \in T} p(h,t) f_j(h,t)$

2. $\tilde{E} f_j = \sum_{i=1}{}^{n} \tilde{p}(h_i, t_i) f_j(h_i, t_i)$

This model can also be interpreted under the Maximum Entropy formalism, in which the goal is to maximize the entropy of a distribution subject to certain constraints. Here, the entropy of the distribution p is defined as follows:

(18) $H(p) = -\sum_{h \in H, t \in T} p(h,t) \log p(h,t)$

During the test step, the tagging procedure gives for each word a list of Y highest probability sequences up to and including that word. The algorithm is a beam search in that for the current word, *only* the Y highest probability sequences up to that point are kept. In calculating sequence probabilities, the algorithm considers every tag for a word, unless it has access to a tag dictionary, in which case, it only considers the tags given for a word in the dictionary. Using this model, (Ratnaparkhi 1996) obtains an accuracy of 96.43% on English test data.

The complexity of the searching procedure for MaxEnt is $O(N_{test} * T * F * Y)$ where N_{test} is the test data size (number of words), T is the number of meaningful tags, F is the average number of features that are active for the given event (h, t) and Y is explained above. The cost of parameter estimation is $O(N_{train} * T * F)$, where T, F are defined above and N_{train} is the training data size, in words.

4.9 Memory-based tagging (MBT)

In the memory-based approach to POS tagging (Daelemans et al. 1996; Daelemans et al. 1999) a set of example cases is kept in memory. Each example case consists of a word with preceding and following context, as well as the corresponding category for that word in that context. Thus, training is simply a matter of selecting the size of the context and storing these cases. A new sentence is tagged by selecting for each word in that sentence the most similar case(s) in memory, and extrapolating the categories of the words from these "nearest neighbors". During testing, the distance between each test pattern (i.e., word plus context information) and all training patterns present in memory is computed. A tag from the "closest" training pattern is assigned to the given word in the test data. When a word is not found in the lexicon, its lexical representation is computed on the basis of its form, its context is determined, and the resulting pattern is disambiguated using extrapolation from the most similar cases in an unknown words case base. In each case, the output is a "best guess" of the category for the word in its current context.

Memory-based tagging requires a large training corpus in order to extract a lexicon. For each word the number of times it occurs with each category is recorded. For the task of tagging English, (Daelemans et al. 1996) generate the lexicon based on a 2-million-word training set and test the tagger on 200K test words, getting an score of 96.4%. For tagging Dutch, they use a training corpus of nearly 600K words and test on 100K words from another corpus, obtaining an accuracy of 95.7%. The English tagset used in the experiments contains about 40 tags, whereas the Dutch tagset has 13 tags.

4.10 Decision trees

(Schmid 1994) develops another technique – a probabilistic decision tree tagger known as TreeTagger. TreeTagger is a Markov model tagger which makes use of a decision tree to get more reliable estimates for contextual parameters. So, the determining context for deciding on a tag is the space of the previous n tags (n=2, in the case of a second order Markov model). The methods differ, however, in the way the transition probability $p(t_n|t_{n-2}t_{n-1})$ is estimated. N-gram taggers often estimate the probability using the maximum likelihood principle, as mentioned above. Unlike those approaches, TreeTagger constructs a binary-branching decision tree. The binary tree is built recursively from a training set of trigrams. The nodes of the tree correspond to questions (or tests) about the previous one or two tags. The branches correspond to either a yes or no answer. For instance, a node might be tag_{-2}*=DET?* which asks whether the tag two previous positions away is a determiner. By following the path down to the terminal elements of the tree, one can determine what the most likely tag is. That is, the terminal elements are sets of $(tag, probability)$ pairs.

To construct the tree, all possible tests are compared to determine which tests should be assigned to which nodes. The criterion used to compare the tests is the amount of information gained about the third tag by performing each test. Each node should divide the data maximally into two subsets (i.e., should ask the question which provides the most information about a tagging decision). To do this, a metric of information gain is used. The information gain is maximized, which, in turn, minimizes the average amount of information still needed *after* the decision is made.

Once a decision tree is constructed, it can be used to derive transition probabilities for a given state in a Markov model. As with other probabilistic classifiers utilizing a Markov model, the Viterbi algorithm is used to find the best sequence of tags. With this, and with training the model on 2M words and testing it on 1K words, (Schmid 1994) obtains 96.36% accuracy using the Penn Treebank tagset.

4.11 Comparison of the tagging approaches

- For any given word, only a few tags are possible, a list of which can be found either in a dictionary or through a morphological analysis of the word.
- When a word has several possible tags, the correct tag can generally be chosen from the local context, using contextual rules that define the valid sequences of tags. These rules may be given different priorities so that a selection can be made even when several rules apply.

Chapter 5

Tagset Design and Morphosyntactically Annotated Corpora

5.1 Tags and tagsets

- *(Morphological) tag* is a symbol encoding (morphological) properties of a word.
- *Tagset* is a set of tags.

The size of a tagset depends on a particular application as well as on language properties.

1. Penn tagset: about 40 tags; `VBD` – verb in past tense
2. Czech positional tagset: about 4000 tags; `VpNS---XR-AA---` (verb, participle, neuter, singular, any person, past tense, active, affirmative)

Types of tagsets

There are many ways to classify morphological tagsets. For our purposes, we distinguish the following three types:

1. atomic (*flat* in Cloeren 1993) – tags are atomic symbols without any formal internal structure (e.g., the Penn TreeBank tagset, Marcus et al. 1993).
2. structured – tags can be decomposed into subtags each tagging a particular feature.
   a) compact – e.g., Multext-East (Erjavec 2004, 2009, 2010) or Czech Compact tagsets (Hajič 2004).
   b) positional – e.g., Czech Positional tagset (Hajič 2004)

Atomic Tagset: An example

The Penn Treebank Tagset, described in Marcus et al. 1993, is an example of an atomic system. The Penn Treebank, a corpus of over 4.5 million words of American English, was annotated with this tagset (1989–1992).

The following part-of-speech tags are used in the corpus:

Table 5.1: tbl:ts:penn

CC	Coord. conjunction	RB	Adverb
CD	Cardinal number	RBR	Adverb, comparative
DT	Determiner	RBS	Adverb, superlative
EX	Existential there	RP	Particle
FW	Foreign word	SYM	Symbol
IN	Prep. / subord. conj.	TO	to
JJ	Adjective	UH	Interjection
JJR	Adjective, comparative	VB	Verb, base form
JJS	Adjective, superlative	VBD	Verb, past tense
LS	List item marker	VBG	Verb, gerund / present part.
MD	Modal	VBN	Verb, past part.
NN	Noun, singular or mass	VBP	Verb, non-3rd p. sg. pres.
NNS	Noun, plural	VBZ	Verb, 3rd p. sg. pres.
NP	Proper noun, singular	WDT	Wh-determiner
NPS	Proper noun, plural	WP	Wh-pronoun
PDT	Predeterminer	WP	Possessive wh-pronoun
POS	Possessive ending	WRB	Wh-adverb
PRP	Personal pronoun	,	Comma
PRP$	Possessive pronoun	.	Sentence-final punctuation

While even in this tagset, some structure could be found (`JJ` vs. `JJR` vs. `JJS`), it is rather ad-hoc and very limited.

Structured tagsets

Any tagset capturing morphological features of richly inflected languages is necessarily large. A natural way to make them manageable is to use a *structured system.* In such a system, a tag is a composition of tags each coming from a much smaller and simpler atomic tagset tagging a particular morpho-syntactic property (e.g., gender or tense).

1. Positional tagsets

    - Tags are sequences of values encoding individual morphological features.
    - All tags have the same length, encoding all the features distinguished by the tagset.
    - Features not applicable for a particular word have a N/A value.
    - E.g., `AAFS4----2A----` (Czech Positional Tagset) encodes adjective (`A`), feminine gender (`F`), singular (`S`), accusative (`4`), comparative (`2`).

2. Compact tagset

- Tags are sequences of values encoding individual morphological features.
- In a compact tagset, the N/A values are left out.
- E.g., `AFS42A` (Czech Compact Tagset) encodes adjective (`A`), feminine gender (`F`), singular (`S`), accusative (`4`), comparative (`2`).

For large tagsets, a structured system has many practical benefits:

1. *Learnability:* It is much easier to link traditional linguistic categories to the corresponding structured tag than to an unstructured atomic tag. While it takes some time to learn the positions and the associated values of the Czech Positional Tagset, for most people, it is still far easier than learning the corresponding 4000+ tags as atomic symbols.
2. *Systematic description:* The morphological descriptions are more systematic. In each system, the attribute positions are (roughly) determined by either POS or SubPOS. Thus, for example, knowing that a token is a common noun (`NN`) automatically provides information that the gender, number, and case positions should have values.
3. *Decomposability:* The fact that the tag can be decomposed into individual components has been used in various applications. For instance, the tagger of (Hajič and Hladká 1998), for a long time the best Czech tagger, operates on the subtag level.
4. *Systematic evaluation:* The evaluation of tagging results can be done in a more systematic way. Each category can be evaluated separately on each morphological feature. Not only is it easy to see on which POS the tagger performs the best/worst, but it is also possible to determine which individual morphological features cause the most problems.

It is also worth noting that it is trivial to view a structured tagset as an atomic tagset (e.g., by assigning a unique natural number to each tag), while the opposite is not true.

Structured tagsets: Examples

MULTEXT-East Tagset

- Originates from EU MULTEXT (Ide and Véronis 1994)
- MULTEXT-East V.1 developed resources for 6 CEE languages as well as for English (the "hub" language)
- MULTEXT-East V. 4 (Erjavec 2010): 13 languages: English, Romanian, Russian, Czech, Slovene, Resian, Croatian, Serbian, Macedonian, Bulgarian, Persian, Finno-Ugric, Estonian, Hungarian.
- MULTEXT specifications are interpreted as feature structures, where a feature-structure consists of a set of attribute-value pairs, e.g., there exists, for Nouns, an attribute *Type*, which can have the values *common* or *proper*. A morpho-syntactic description (MSD) (=tag) corresponds to a fully specified feature structure.
- Compact:
  - Positions' interpretations vary across different parts of speech. For instance, for nouns, position 2 is Gender, whereas for verbs, position 2 is VForm, whose meaning roughly corresponds to the mood.

  – e.g., `Ncmsn` (noun, common, masculine, singular, nominative); `Ncmsa--n` (noun, common, masculine, singular, accusative, indefinite, no clitic, inanimate. A number of Slavic languages are sensitive to animacy, i.e., nouns decline differently depending on their animacy. So, in the former (compact) tag, the animacy specification is irrelevant and therefore, is omitted.

CLiC-TALP

- CLiC-TALP (Civit 2000) tagsets were developed for Spanish and Catalan.
- The Spanish CLiC-TALP system is a structured system, where the attribute positions are determined by POS.
- The tagset distinguishes 13 parts of speech (POS) categories.
- It also makes more fine-grained morphological distinctions for mood, tense, person, gender, number, etc., for the relevant categories.
- Tag size: 285.
- E.g., `AQ0CS0` 'rentable' (adjective, qualitative, inapplicable case, common gender, singular, not a participle).
- Uses the ambiguous `0` value for a number of attributes – It can sometimes mean "non-applicable" and sometimes "null".

Czech PDT

One of the good representative of the positional tag system is the Prague Dependency Treebank (PDT) Tagset (`http://ufal.mff.cuni.cz/pdt`). Its basic features, as outlined above, are:

1. The first position specifies POS.
2. The second position specifies Detailed POS (SubPOS).
3. SubPOS uniquely determines POS.
4. SubPOS generally determines which positions are specified (with very few exceptions).
5. The `-` value meaning N/A or not-specified is possible for all positions except the first two (POS and SubPOS).

Thus, unlike what we find in the Multext-East tagsets, the position of a particular attribute is the same regardless of the POS. If it is inappropriate for a particular POS (or more precisely SubPOS), it simply has a N/A value (`-`).

The Czech tagset uses a rather large number of wildcards, i.e., values that cover more than one atomic value. For example, consider gender, as Figure 5.1 shows there are four atomic values, and six wildcard values, covering not only various sets of the atomic values (e.g., `Z` = {`M`,`I`,`N`) , but in one case also their combination with number values (`QW` = {`FS`,`NP`}).

On the other hand, there are some values appropriate for a single word. For example, a tag with subPOS value `E`, whose only member is the relative pronoun *což*, which corresponds to the English *which* in subordinate clauses.

It is worth noting, that the values of detailed part of speech do not always encode the same level of detail. If the values are seen as a separate tagset, it is an atomic tagset which could be

Table 5.2: Positional Tag System for Czech

Position	Abbr	Name	Description	Example *viděl*o ‘saw’	
1	p	POS	part of speech	`V`	verb
2	s	SubPOS	detailed part of speech	`p`	past participle
3	g	gender	gender	`N`	neuter
4	n	number	number	`S`	singular
5	c	case	case	`--`	n/a
6	f	possgender	possessor’s gender	`--`	n/a
7	m	possnumber	possessor’s number	`--`	n/a
8	e	person	person	`X`	any
9	t	tense	tense	`R`	past tense
10	d	grade	degree of comparison	`--`	n/a
11	a	negation	negation	`A`	affirmative
12	v	voice	voice	`A`	active voice
13		reserve1	unused	`--`	n/a
14		reserve2	unused	`--`	n/a
15	i	var	variant, register	`--`	basic variant

Atomic values:		
`F`		feminine
`I`		masculine inanimate
`M`		masculine animate
`N`		neuter
Wildcard values:		
`X`	`M`, `I`, `F`, `N`	any of the basic four genders
`H`	`F`, `N`	feminine or neuter
`T`	`I`, `F`	masculine inanimate or feminine (plural only)
`Y`	`M`, `I`	masculine (either animate or inanimate)
`Z`	`M`, `I`, `N`	not feminine (i.e., masculine animate/inanimate or neuter)
`Q`		feminine (with singular only) or neuter (with plural only)

Figure 5.1: Atomic and wildcard gender values

naturally expressed as a structured tagset having two positions expressing two levels of detail. For example, there is no single value encoding personal pronouns. Instead, there are three values encoding three different types of personal pronouns: `P` (regular personal pronoun), `H` (clitical personal pronoun), and `5` (personal pronoun in prepositional form). Similarly, there are eight values corresponding to relative pronouns, four to generic numerals, etc.

5.2 Tagset size and tagging accuracy

- Tagsets for highly inflected languages are typically far bigger that those for English.
- It might seem obvious that the size of a tagset would be negatively correlated with tagging accuracy: for a smaller tagset, there are fewer choices to be made, thus there is less opportunity for an error.
- Elworthy (1995) shows, this is not true.

Let's assume a language where determiners agree with nouns in number, determiners are non-ambiguous for number while nouns sometimes are. Consider two tagsets: one containing four tags, singular determiner, plural determiner, singular noun, plural nouns; and another containing three tags, determiner, singular noun, plural noun. A bigram tagger will get better accuracy when using the larger tagset: Since determiners are non-ambiguous, a determiner is tagged correctly and it in turn determines the correct tag for the noun. In the smaller tagset, the determiner is also tagged correctly; however, the tag does not provide any information to help in tagging the noun.

5.3 Harmonizing tagsets across languages?

- Pros:
  - Harmonized tagsets make it easier to develop multilingual applications or to evaluate language technology tools across several languages.
  - Interesting from a language-typological perspective as well because standardized tagsets allow for a quick and efficient comparison of language properties.
  - Convenient for researchers working with corpora in multiple languages – they do not need to learn a new tagset for each language.
- Cons:
  - Various grammatical categories and their values might have different interpretations in different languages.

    For example, the notion of definiteness is expressed differently in various languages. In English, definiteness is expressed using the definite determiners; in Romanian, in turn, definite clitics attach to the end of nouns; in Lithuanian, in turn, morphologically expressed definiteness exists only in pronominal adjectives.

    Another example is the category of plural. In Russian and Slovenian, for example, the plural value does not mean the same thing: in the former, there is only singular vs. plural dichotomy, in the latter, however, there is also the dual grammatical number, which is a separate form of every noun used when there are only two such items.

5.4 Summary: Tagset design challenges

We have discussed a number of tag systems and outlined several important consideration when designing a tagset. These included questions like

- Tagset size: computationally tractable? Linguistically adequate?
- Atomic or Structural? If Structural, compact or positional?
- What linguistic properties are relevant?
  - For instance, the Czech tagset mixes the morpho-syntactic annotation with what might be called dictionary information, e.g., the gender of information for nouns is included in the tag.

  - The Czech tagset sometimes combines several morphological categories into one.
  - The Czech tagset sometimes creates many additional categories, whose members are singletons:

    E.g., verbs, cardinal numerals and certain pronouns have the value H for gender, if they are either feminine or neuter; or participles, nominal forms of adjectives and verbs are tagged as `Q` for the gender position, if they are feminine (singular) or neuter (plural).

    On the other hand, there is a subpos value `E`, whose only member is the relative pronoun *což*, which corresponds to the English *which* in subordinate clauses. Other relative pronouns belong to other separate categories, such as `J` (=relative pronouns not after a preposition) and `9` (=relative pronouns after a preposition).

- Should the system be standardized and be easily adaptable for other languages?

Chapter 6

Unsupervised and Resource-light Approaches to Computational Morphology

Recently, there has been an increased interest in statistical modeling of morphology and of morphological induction and in particular the unsupervised or lightly supervised induction of morphology from raw text corpora. This work, while impressive, is still not at the stage where one can induce a morphological analyzer such as Koskenniemi's system for Finnish (see §3.3) The statistical approaches address the issue of finding simple relations between morphologically related words, involving one or two affixes.

Before we proceed further, we want to note that even though we do try to compare various morphological analyzers, the task is subjective. There are many corpora where each word is annotated with the lemma and tag appropriate in a given context. Such corpora are suitable for evaluating taggers. However, for the evaluation of morphological analyzers, the annotation should contain *all* morphologically plausible analyses regardless of context. And there are not many corpora like that.

The discussion below is partly based on (Roark and Sproat 2007).

6.1 Linguistica (Goldsmith 2001)

One of the most cited systems for automatic unsupervised morphological acquisition is Linguistica. The system is available online: `http://linguistica.uchicago.edu/`.

- The goal of the algorithm is to learn affixation alternations;
- The system starts with an unannotated corpus of text of a language and derives a set of *signatures* along with words that belong to those signatures;
  - Signatures are sets of suffixes that are used with a given set of stems. See Table 6.1 for examples.
  - Signatures are NOT paradigms: 1) They can contain both derivational and inflectional affixes; 2) the set is not complete (e.g., the *-ed* is missing), but it might show up in other signatures.

  - Going from signatures to paradigms is not trivial. The system is not capable of handling some alternations, such as *blow/blew* since the method only handles suffixation and does not consider any phonological/graphemic alternations. Thus ending/suffix is the part of the word than changes and may include part of the previous morpheme. In this respect, the system is similar to (Hajič 2004) discussed in §3.

Table 6.1: Example of signatures

NULL.ed.ing	betray, betrayed, betraying
NULL.ed.ing.s	remain, remained, remaining, remains
NULL.s	cow, cows
e.ed.ing.es	notice, noticed, noticing, notices

Deriving signatures

- Step 1: Derive candidate signatures and signature-class membership.
- Step 2: Evaluate the candidate.

Step 1: Candidate generation

Word segmentation

- Generate a list of potential affixes:
  - Start at the right edge of each word in the corpus,
  - Collect the set of possible suffixes up to length six,
  - For each of these suffixes, compute the following metric (where N$_k$ is the total number of k-grams):
    $\frac{freq(n_1,n_2...n_k)}{N_k} log \frac{freq(n_1,n_2...n_k)}{\prod 1^k freq(n_i)}$
- The first 100 top ranking candidates are chosen on the basis of a Boltzmann distribution; and words in the corpus are segmented according to these candidates.
- Suffixes that are not optimal for at least one word are discarded.
- **Output**: a set of stems and associated suffixes including the null suffix. The alphabetized list of suffixes associated with each stem constitutes the signature for that stem.
- Simple filtering: remove all signatures associated only with one stem or only with one suffix. See examples in Table 6.1.

Step 2: Candidate evaluation

- Not all suggested signatures are useful. They need to be evaluated.
- Evaluate candidates using minimum description length (MDL; Rissanen 1989); see also Kazakov 1997; Marcken 1995):

  - MDL is based on the insight that a grammar can be used to compress a corpus; the better the morphological description is, the better the compression is. MDL considers sum of the size of the grammar and of the size of the compressed corpus. This is a standard measure in text compression: a good compression algorithm is one that minimizes the size of the compressed text plus the size of the model that is used to encode and decode that text.

- The compressed length of the model – the morphology – is given by:

  $\lambda < T > + \lambda < F > + \lambda < \Sigma >$

  where, $\lambda < T >$ represents the length (in bits) of a list of pointers to ¡T¿ stems, where T is the set of stems, and the notation $<>$ reoresents the cardinality of that set. $\lambda < F >$ and $\lambda < \Sigma >$ represent the equivalent pointer-list lengths for suffixes and signatures, respectively.

  - That's how these expressions are calculated:

    $\lambda < T >= \Sigma_{t \in T}(log(26) * length(t) + log\frac{[W]}{[t]})$,

    where $[W]$ is the number of word tokens in the corpus and $[t]$ is the number of tokens of the paticular t. The $log(26)$ term assumes an alphabet of 26 letters.

    $\lambda < F >= \Sigma_{f \in suffixes}(log(26) * length(f) + log\frac{[W_A]}{[f]})$,

    where $[W_A]$ is the number of tokens of morphologically analyzed word, and $[f]$ is the number of tokens of the suffix f.

    $\lambda < \Sigma >= \Sigma_{\sigma \in \Sigma} log\frac{[W]}{[\sigma]}$,

    where Σ is the set of signatures.

    The compressed length of the corpus in terms of the morphological model is calculated in the following way:

    $\Sigma_{w \in W}[w][log\frac{[W]}{[\sigma(w)]} + log\frac{[\sigma(w)]}{[stem(w)]} + log\frac{[\sigma(w)]}{[suffix(w) \in \sigma(w)]}]$

    Assuming the maximum likelihood estimate for probabilities:

    $\Sigma_{w \in W}[w][-log(P(\sigma(w))) - log(P(stem(w)|\sigma(w))) - log(P(suffix(w)|\sigma(w)))]$

Evaluation Goldsmith tested his method on English, French, Italian, Spanish, and Latin. Having no gold standard against which to compare, he evaluated the results subjectively, classifying the analyses into the categories *good, wrong analysis, failed to analyze, spurious analysis.* For English, for example, the results were 82.9% in the *good* category, with 5.2% wrong, 3.6% failure, and 8.3% spurious.

Problems

- Analyzes only suffixes (can be generalized to prefixes as well). Handling stem-internal changes would require significant overhaul.
- All phonological/graphemic changes accompanying inflection, must be factored into suffixes: e.g., Russian *pla**k**at'* (cry.INF) and *pla**č**et* (cry.Pres.3P); It would be also hard to tell that *hated* should not be analyzed as *hat+ed.*
- Ignores syntax and semantics. It has been demonstrated that syntactic and semantic information does indeed help with the acquisition of morphology. Without semantic information, it would be hard to tell that *ally* should not be analyzed as *all+y.* For example, (Schone and Jurafsky 2000) use semantic, orthographic, and syntactic information derived from unannotated corpora to arrive at an analysis of inflectional morphology.

6.2 Yarowsky & Wicentowski 2000

Yarowsky and Wicentowski (2000) present an algorithm for a resource-light induction of inflectional paradigms (suffixal and irregular). They test their approach on induction of English present-past verb pairs. They discover forms of the same paradigm in a large unannotated corpus using a combination of four similarity measures:

1. expected frequency distributions,
2. context,
3. weighted Levenshtein distance,
4. an iteratively bootstrapped model of affixation and stem-change probabilities.

They divide this task into three separate steps:

1. Estimate a probabilistic alignment between inflected forms and base forms in a given language.
2. Train a supervised morphological analysis learner on a weighted subset of these aligned pairs.
3. Use the result in Step 2 as either a stand-alone analyzer or a probabilistic scoring component to iteratively refine the alignment in Step 1.

The morphological induction assumes the following available resources:

1. List of inflectional categories, each with canonical suffixes.
2. A large unannotated text corpus.
3. A list of the candidate noun, verb, and adjective base forms (typically obtainable from a dictionary)
4. A rough mechanism for identifying the candidate parts of speech of the remaining vocabulary, not based on morphological analysis (e.g., .
5. A list of consonants and vowels.
6. Optionally, a list of common function words.
7. Optionally, various distance/similarity tables generated by the same algorithm on previously studied languages can be useful as seed information, especially if these languages are closely related.

Alignment by frequency similarity. This measure assumes two forms belong to the same lemma, when their relative frequency fits the expected distribution. The distribution of irregular forms is approximated by the distribution of regular forms.

This measure worked well for verbal tense, but it would have to be modified to handle categories where one can expect multimodal distribution. For example, consider the number of nouns: the distribution is different for count nouns, mass nouns, plurale-tantum nouns, etc.

Alignment by context similarity. This measure is based on the idea that inflectional forms of the same lemma have similar selectional preferences (mostly much closer than even synonyms). For example, related verbs tend to occur with similar subjects/objects. To minimize needed training resources, Yarowsky and Wicentowski 2000 identify the positions of head-noun objects and subjects of verbs using a set of simple regular expressions. The authors notice that such expressions extract significant noise and fail to match many legitimate contexts, but because they are applied to a large monolingual corpus, the partial coverage is tolerable.

This measure worked well for verbs, but the question is how it would perform for other parts-of-speech, where the subcategorization requirements are much less strict?

Alignment by weighted Levenshtein distance. This similarity measure considers overall stem edit distance using a weighted Levenshtein measure (Levenshtein 1966). One important feature of this distance measure is that the edit costs for vowels and consonants are not the same. The motivation for the difference in costs stems from the assumption that in morphological systems worldwide, vowels and vowel clusters tend to change more often during inflection than consonants. Rather than treating all string edits as equal, four values are used: V for vowels, V+ for vowel clusters, C for consonants, and C+ for consonant clusters. They are initially set to relatively arbitrary assignments reflecting their respective tendencies towards mutability, and then are iteratively re-estimated. A table from a similar language can also be used to set the initial edit costs. Even though this approach is shown to work for several languages , there is no linguistic research that supports this claim.

Alignment by a generative probabilistic model. This alignment is done with morphological transformation probabilities. The goal is to generalize the inflection-root alignments via a generative probabilistic model. At each iteration of the algorithm, the probabilistic mapping function is trained on the table output of the previous iteration (i.e., on the root-inflection pairs with optional POS tags, confidence scores, and stem change+suffix analysis). Each training example is weighted with its alignment confidence, and mappings which have low confidence are filtered out.

Of the four measures, no single model is sufficiently effective on its own. Therefore, traditional classifier combination techniques are applied to merge scores of the four models.

Problems Applying the method developed by Yarowsky and Wicentowski 2000 to languages used in the current context raises a number of problems.

- The suffix-focused transformational model is not sufficient for languages such as Russian that exhibit prefixal morphology.[1]
- Most of the difficult substance of the lemmatization problem is often captured in Yarowsky and Wicentowski's 2000 work by a large *root+POS↔inflection* mapping table and a simple transducer to handle residual forms. Unfortunately, such an approach is not directly applicable to highly inflected languages, such as Czech or Russian, where sparse data becomes an issue.
- (Yarowsky and Wicentowski 2000) use the Cucerzan and Yarowsky's 2002 bootstrapping approximation of tag probability distributions. Their algorithm starts with a small annotated corpus. For French, for example, the initial training data was 18,000

[1]The morphological analyzer used in the experiments in subsequent chapters does not handle prefixes either, except for the negative *ne-* and the superlative *nai-*.

tokens. Here, the goal is to develop a portable system which will use much smaller, if any, training corpus of the target language. Moreover, manually creating an annotated corpus that uses such fine-grained morpho-syntactic descriptions is extremely time-consuming.

Even though the algorithm described by (Yarowsky and Wicentowski 2000) cannot be used directly because of the issues outlined above, their ideas, to a large extent, inspired the current work. The main goal here is to produce detailed morphological resources for a variety of languages without relying on large quantities of annotated training data. Similarly to their work, our work relies on a subset of manually encoded knowledge, instead of applying completely unsupervised methods.

6.3 Unsupervised taggers

As mentioned above, the problem with using supervised models for tagging resource-poor languages is that supervised models assume the existence of a labeled training corpus. Unsupervised models do not make this assumption, which makes them more applicable to the task of morpho-syntactic tagging resource-poor languages.

Unsupervised models generally rely on the presence of a dictionary, or lexicon, which contains the possible parts of speech for a given word type. This list of parts of speech may be ordered or unordered and in the former case may contain probabilities. For each word token in the corpus, the parts of speech in the dictionary for that word type are considered as possibilities in tagging.

Markov models

MM taggers work well when there is a large, tagged training set. MMs can be used without a corpus to train on, too. In the unsupervised case, the MM approach (Cutting et al. 1992; Jelinek 1985; Merialdo 1994) still has three major components: 1) an initial (probability) vector, 2) a transition (probability) matrix, and 3) an emission (probability) matrix. Each of these components are iteratively estimated until the process converges. For tagging, the Viterbi algorithm is used, as described in §4.4.

The difference between Visible MM (VMM) tagging (i.e., supervised) and Hidden MM (HMM) tagging (i.e., unsupervised) is in how the model is trained. Since no pre-tagged corpus is available, the probabilities have to be estimated in some other way. To do this the initial parameters of the model are set based on a dictionary that lists all possible tags for each word.

There are two steps in HMM training — expectation (estimation) and maximization, which alternate during the training process, thus giving the Expectation Maximization (EM) algorithm[2]. Basically, first the parameters of the model are estimated — the initial, transition, and emission probabilities — and then the Viterbi algorithm is used to determine which estimation maximizes the probability of a sequence of tags. This sequence of tags is then used to reestimate the parameters.

[2]The *Baum-Welch* or *Forward-Backward* algorithm, which is used for HMM training, is a special case of general EM.

When the probability of traversing an arc from t_i to t_{i+1} is estimated. Both forward probabilities (probability of the sequence of tags leading up to t_i) and backward probabilities (probability of the sequence of tags following t_{i+1}) are examined. During the expectation phase, a forward pass over the data is made to (re)-estimate the forward probabilities and a backward pass for backward probability (re)-estimation. This multi-directional information gives a better estimate of the probability of traversing an arc than can be obtained using forward probabilities alone.

With an unsupervised HMM tagger, Cutting et al. 1992 are able to obtain accuracies of up to 96% for English, on par with other current technologies. This raises the question whether such an approach could be used for other languages.

Transformation-based learning (TBL)

In the supervised transformation-based learning (TBL), a corpus is used for scoring the outcome of applying transformations in order to find the best transformation in each iteration of learning. In the unsupervised case, this scoring function must be found without a manually tagged corpus. To adapt to a new scoring function, Brill 1995, 1999 redefines all three components of the TBL model.

The unsupervised TBL learner begins with an unannotated text corpus, and a dictionary listing words and the allowable part of speech tags for each word. The initial state annotator tags each word in the corpus with a list of all allowable tags.

Since now instead of sets of tags, one tag per word is used, the transformation templates must also be changed. Instead of being templates which change one tag to another, they select a tag from the set of tags. That is, they change a word's tagging from a set of tags to a single tag. A template for such transformations is as outlined in (19). The context C can be defined as before, although Brill 1999 limits the context to the previous (following) word/tag.

(19) Change the tag of a word from χ to Y in context C.

where χ is a set of two or more tags and Y is a single tag, such that $Y \in \chi$.

When using supervised TBL to train a POS tagger, the scoring function is just the tagging accuracy that results from applying a transformation. With unsupervised learning, the learner does not have a gold standard training corpus with which accuracy can be measured. Instead, the information from the distribution of unambiguous words is used to find reliable disambiguating contexts.

In each learning iteration, the score of a transformation is computed based on the current tagging of the training set. As stated above, each word in the training set is initially tagged with all tags allowed for that word, as indicated in the dictionary. In later learning iterations, the training set is transformed as a result of applying previously learned transformations.

To calculate the score for a transformation rule, as described in (19), Brill computes (20) for each tag $Z \in \chi, Z \neq Y$.

(20) $freq(Y)/freq(Z) * incontext(Z, C)$,

where $freq(Y)$ is the number of occurrences of words unambiguously tagged with tag Y in the corpus, $freq(Z)$ is the number of occurrences of words unambiguously tagged with tag Z in the corpus, and the $incontext(Z, C)$ is the number of times a word unambiguously tagged with tag Z occurs in context C in the training corpus. To produce a score, first let R be defined as in (21). Then the score for the transformation in (19) is as in (22).

(21) $R = argmax_Z \, freq(Y)/freq(Z) * incontext(Z, C)$

(22) $incontext(Y, C) - freq(Y)/freq(R) * incontext(R, C)$

To further explain what the scoring function in (22) does, first consider that a good transformation for removing the tag ambiguity of a word is one for which one of the possible tags appears much more frequently. This is measured here by unambiguously tagged words in the context, after adjusting for the differences in relative frequency between different tags (i.e., $freq(Y)/freq(R)$). So, the comparison is made between how often Y unambiguously appears in a given context C and the number of unambiguous instances of the most likely tag R in the same context, where $R \in \chi, R \neq Y$. The tag is changed from χ to Y, if Y is the best choice. That is, the learner will accept a transformation for a given learning iteration if the transformation maximizes the function in (22).

Chapter 7

Our Approach to Resource-light Morphology

In this section we address the development of taggers for resource-poor languages. We describe a rapid, low-cost approach to the development of taggers by exploring the possibility of approximating resources of one language by resources of a related language. Our approach takes the middle road between knowledge-free approaches and those that require extensive manually created resources. We believe that for many languages and applications, neither of these extreme approaches is warranted. The knowledge-free approach lacks precision and the knowledge-intensive approach is usually too costly.

Our main assumption is that a model for the target language can be approximated by language models from one or more related source languages and that inclusion of a limited amount of high-impact and/or low-cost manual resources is greatly beneficial and desirable. Our research has already given positive and promising results (especially Feldman and Hana 2010, but also Feldman 2006; Feldman et al. 2006; Hana et al. 2004; Hana 2008; Hana et al. 2006).

For expository reasons, below, we most concentrate on the Russian-Czech pair. We have successfully tested our approach by creating taggers for other languages as well (e.g., Portuguese and Catalan). We use TnT (Brants 2000), a second order Markov Model tagger. The language model of such a tagger consists of emission probabilities (corresponding to a lexicon with usage frequency information) and transition probabilities (roughly corresponding to syntax rules with strong emphasis on local word-order). We approximate the target-language emissions by combining the emissions from the (modified) source language corpus with information from the output of our resource-light analyzer (Hana 2008). The target-language transitions are approximated by the source language transitions (Feldman and Hana 2010). We also need to account for the fact that the two languages have different (although often related) tagsets. Below, we describe our approach in more detail.

We experimented with several language pairs.

- Russian via Czech
- Catalan via Spanish
- Portuguese via Spanish

Below we mostly concentrate on Russian via Czech, but we do mention the other language pairs and the result of the experiments.

7.1 Tagsets

For all languages in the experiments, we used positional tagsets. The advantages of positional tagsets were outlined in Chapter 5.

All tagsets follow the basic design features of the Czech positional tagset (Hajič 2004):

1. The first position specifies POS.
2. The second position specifies Detailed POS (SubPOS).
3. SubPOS uniquely determines POS.
4. SubPOS generally determines which positions are specified (with very few exceptions).
5. The `-` value meaning N/A or not-specified is possible for all positions except the first two (POS and SubPOS).

Czech

In the Czech positional tag system, every tag is represented as a string of 15 symbols, each corresponding to one morphological category. See §5.1 for more details.

Russian

The Russian tagset (Hana and Feldman 2010) we use was developed on the basis of the Czech positional tagset. The tagsets encode similar set of morphological categories in the same order and in most cases do so using the same symbols. However, there are some differences. Many of them are a consequence of linguistic differences between the languages. For example, Russian has neither vocative nor dual, nor does it have auxiliary or pronominal clitics; and the difference between colloquial and official Russian is not as systematic and profound as in Czech. Table 7.1 compares the number of values for individual positions.

The Russian tagset also uses far fewer wildcards (symbols representing a set of atomic values). Even though wildcards might lead to better tagging performance, we intentionally avoid them. The reason is that they provide less information about the word, which might be needed for linguistic analysis or an NLP application. In addition, it is trivial to translate atomic values to wildcards if needed.

The Russian tagset contains only wildcards covering all atomic values (denoted by `X` for all applicable positions). There are no wildcards covering a subset of atomic values. Forms that would be tagged with a tag containing a partial wildcard in Czech are regarded as ambiguous.

For example, where the Czech tagset uses `Z` (all genders except feminine), our Russian tagset uses `M` (masculine) or `N` (neuter) depending on the context. Thus, Czech *tomto* ‘this$_{masc/neut.loc}$’ is tagged as `PDZS6----------` in *v tomto domě* ‘in this house$_{masc}$’ and *v tomto místě* ‘in this place$_{neut}$’, while Russian *ètom* ‘this$_{masc/neut.loc}$’ is tagged as `PDMXS6----------` in *v ètom dome* ‘in this house$_{masc}$’ and `PDNXS6----------` in *v ètom meste* ‘in this place$_{neut}$’.

Table 7.1: Comparison with the Czech Positional Tagset

Rus	Cze	Abbr	Name	No. of values	
				Czech	Russian
1	1	p	Part of Speech	12	12
2	2	s	SubPOS (Detailed Part of Speech)	69	43
3	3	g	Gender	11	4
4		y	Animacy	6	4
5	4	n	Number	9	4
6	5	c	Case	5	8
7	6	f	Possessor's Gender	5	5
8	7	m	Possessor's Number	3	3
9	8	e	Person	5	5
10		r	Reflexivity		3
11	9	t	Tense	5	5
12		b	Verbal aspect		4
13	10	d	Degree of comparison	4	4
14	11	a	Negation	3	3
15	12	v	Voice	3	3
	13		Not used	1	
	14		Not used	1	
16	15	i	Variant, Abbreviation	10	8

Romance tagsets

We used positional tags for the Romance languages as well. For Spanish and Catalan we translated the structured tags, provided by the CLiC-TALP project (`http://clic.ub.edu/en/what-is-clic`) into our positional system. The Portuguese tagset was developed from scratch. The reader interested in the details of the Slavic and Romance tagsets is referred to Feldman and Hana 2010.

Table 7.2: Overview and comparison of the Romance tagsets

Pos	Description	Abbr.	No. of values		
			Spanish	Portuguese	Catalan
1	POS	p	14	14	14
2	SubPOS – detailed POS	s	29	30	29
3	Gender	g	6	6	6
4	Number	n	5	5	5
5	Case	c	6	6	6
6	Possessor's Number	m	4	4	4
7	Form	o	3	3	3
8	Person	e	5	5	5
9	Tense	t	7	8	7
10	Mood	m	7	7	7
11	Participle	r	3	3	3

Table 7.3: Overview of the tagsets we use

Language	size	# of tags in 1,893word-corpus	# of positions
Czech	4,251	216	13 (+2 not used)
Russian	1,063	179	13 (+2 not used)
Spanish	282	109	11
Catalan	289	88	11
Portuguese	259	73	11

7.2 Corpora

Since we wanted to stay within the resource-light paradigm, we intentionally avoided the use of parallel corpora or target-language annotated corpora. For Russian, Czech, and Catalan, we used a small annotated development corpus (`Dev`) of around 2K tokens to tune our tools. For Portuguese, unfortunately, such a corpus was not available to us. To evaluate the performance of the system, we always tried to obtain the largest test corpus available.

We used the following corpora for each target language (Czech, Russian, Catalan, and Portuguese):

1. `Dev` – annotated corpus, about 2K (intentionally small).

   For development testing, testing of hypotheses and for tuning the parameters of our tools; not available for Portuguese.

2. `Test` – annotated corpus, preferably large.

   For final testing.

3. `Raw` – raw unannotated corpus, no limit on size.

   Used in cognate detection (see §7.6), to acquire lexicon, to get the most frequent words.

4. `Train` – large annotated corpus.

   Used to report statistics (not used during development); available only for Czech and Catalan.

and the following corpora for each source language (Czech and Spanish):

1. `Train` – large annotated corpus.

   Used to train the source language tagger (i.e., emission and transition probabilities) and to report statistics.

2. `Raw` – large unannotated corpus.

   Used in cognate detection, to get the most frequent words.

Table 7.4 summarizes the properties of the corpora we used in our experiments. For each target and source language, we report the size of the training, development, and test corpora, the sources we used, the type of the tagset, and whether a corpus was annotated manually or automatically. Thus, for example Russian `Dev` and `Test` and Portuguese `Test` were annotated manually. The term *positionalized* means that we translated a tagset into our positional system.

The Russian and Portuguese corpora were annotated by us. The annotation process and supporting tools are described in §8.6 below.

Table 7.4: Overview of the corpora

Language	Corpus	Size	Source	Manual/Automatic tagging	Tagset
Czech (src/target)	`Dev`	2K	PDT 1.0	Manual	Czech Positional
	`Test`	125K	PDT 1.0	Manual	Czech Positional
	`Raw`	39M	distributed w/ PDT 1.0	N/A	–
	`Train`	1.5M	PDT 1.0	Manual	Czech Positional
Russian (target)	`Dev`	1,758	Orwell's 1984	Manual (by us)	Russian Positional
	`Test`	4,011	Orwell's 1984	Manual (by us)	Russian Positional
	`Raw`	1M	Upsalla	N/A	–
Spanish (src)	`Train`	106K	CLiC-TALP	Automatic, manually validated	positionalized CLic-TALP
Catalan (target)	`Dev`	2K	CLiC-TALP	Automatic, manually validated	positionalized CLic-TALP
	`Test`	20.6K	CLiC-TALP	Automatic, manually validated	
	`Raw`	63M	El Periodico	N/A	–
	`Train`	80.5K	CLiC-TALP	Automatic, manually validated	positionalized CLic-TALP
Portuguese (target)	`Test`	1,893	NILC	Manual (by us)	modified CLic-TALP
	`Raw`	1.2M	NILC	N/A	

7.3 Experiments: An Overview

In the following, we only discuss one pair of languages (Russian via Czech), but similar experiments were run for other language pairs. We treat Czech as an approximation of Russian. In the simplest model (see §7.4), we use a Czech tagger to tag Russian directly (modulo tagset and script mappings).

In the subsequent experiments, we improve this initial model (making sure we stay in the labor- and knowledge-light paradigm):

1. We use the morphological analyzer (MA) from §7.5 to approximate emissions (§7.5).
2. We use an automatically acquired list of cognates to combine Czech emissions and emissions based on the MA (§7.6).
3. We apply simple syntactic transformations to the Czech corpus ("Russifications") to make it more Russian-like and thus improving the acquired transitions (§7.7).
4. We train batteries of taggers on subtags to address the data sparsity problem (§7.8).

7.4 Experiment 1: Direct tagging with the source-language model

We show that the transition information acquired from the source language, Czech, is also useful for the related target language, Russian. In this model we assume that Czech is such a good approximation of Russian, that we can use it to tag Russian directly (modulo tagset and script mappings).

7.5 Experiment 2: Approximating emissions with morphological analysis

We have approximated the target-language emissions by combining information from the output of our resource-light morphological analyzer and emission probabilities from the source language corpus. In order to explain how it was done, we have to digress and describe our resource-light morphological analyzer.

Resource-light Morphological Analyzer

We have built an open, modular and fast system for morphologically analyzing fusional languages (Hana et al. 2004; Hana 2008). Our system's precision is close to that of a supervised system, but is far less labor-intensive.

Motivation To motivate our approach, we provide some facts about Czech nouns, assuming other open classes and other fusional languages behave similarly. The statistics are based on a subcorpus of the Prague Dependency Treebank (PDT; Bémova et al. 1999). It contains about 220,000 noun tokens corresponding to about 24,000 lemmas. Table 7.5 break lemmas into deciles by their frequency and compares their corpus coverage. Similar to Zipf's law (Zipf 1935, 1949), it makes two things apparent:

Lemma freq decile	Number of tokens	Corpus coverage (%)	Cumulative coverage (%)	Lemmas not in `tr2` (%)
10	164,643	74.1	74	0.2
9	22,515	10.1	84	6.7
8	11,041	5.0	89	22
7	6,741	3.0	92	36
6	4,728	2.1	94	48
5	3,179	1.4	96	61
4	2,365	1.1	97	65
3	2,364	1.1	98	70
2	2,364	1.1	99	75
1	2,364	1.1	100	77

Note: Each decile contains 2364 or 2365 noun lemmas.

Table 7.5: Corpus coverage by lemma frequency

- First, it is easy to get decent coverage of a text by knowing how to analyze a small number of high frequency words – 2.4K of the most frequent lemmas (10th decile) cover 74% of noun tokens in the corpus, and 7.1K lemmas (top 3 deciles) cover nearly 90% of all noun tokens.
- Second, it is very hard, practically impossible, even with very large lexicons to achieve perfect coverage of a running text.
  - The marginal increase in coverage drops sharply. Each of the lower 5 deciles increases the coverage by 1% only, 74times less than the highest decile. This is an expected consequence of the sparse data properties of natural language, but this fact does not free us from the need to handle the phenomenon.
  - Less frequent words tend to be more text specific: 60–77% of the lemmas in each of the lower 5 deciles did not occur in another part of the PDT of the same size – even though both corpora are very similar, having the same newspapers from the same period as sources. Again, this is expected, but must be handled.

Structure

The design of the system allows us to combine modules into a pipeline with different levels of precision. The general strategy is to run "sure thing" modules (those that make fewer errors and overgenerate less) before "guessing" modules that are more error-prone and given to overgeneration. This, for example, means that modules based on manually created resources are assumed reliable and used early in the pipeline, while those that depend on automatically acquired resources are assumed less reliable and used only later. The current system contains the following modules:

- Word list – a list of words, each accompanied with its possible analyses.
- Lexicon-based analyzer. In the lexicon, each lemma is associated with its paradigm and possibly irregular stem(s).
- Guesser – analyzes words relying purely on the analysis of possible endings. In many languages, including the Slavic languages, the situation is complicated by high incidence of homonymous endings – for example the ending *a* has about 19 different meanings in Czech, as Table 2.3 in §2.7 shows. The situation is even more complicated because morpheme boundaries are not known.

  The guesser can optionally use various filters using partial morphological information about words (e.g., lemmas or form with POS only, instead of full tags). In practical settings, such partial information is usually more likely to be available than full analyses.
- Modules for identifying abbreviations, numeric expressions, etc.

In the experiments so far, we have manually provided analyses of the most frequent words, information about paradigms and several derivational affixes (only in experiments with Czech; used in automatic lexicon acquisition; about 10 affixes). The lexicon, list of abbreviation etc. were automatically derived from an unannotated corpus.

Evaluation of the MA

Table 7.6 shows the results of our Czech analyzer on the evaluation data of PDT (about 125K tokens). Our precision is only slightly worse than the current state of the art (Hajič 2004). While the recall of our system is worse, we achieved this results with very little of manually created and thus costly, resources – analyses of 10,000 frequent forms, list of inflectional paradigms and rough information about 20 derivational suffixes. The system of Hajič (2004) uses a manually created lexicon containing about 300,000 entries and a list of paradigms. It is worth stressing that the cost of providing analysis for 10,000 forms is significantly lower than the cost of providing 10,000 lexical entries. The table also lists results (on all tokens) for Russian, Portuguese and Catalan.

Language		Czech (nouns)		Russian	Portuguese	Catalan
		our system	state of the art			
size of manually provided resources	lexicon	0	300K	0	0	0
	word list	10K	0	1K	0	1K
	paradigms	+	+	+	+	+
	derivations	20	?	0	0	0
recall		96.6	98.7	93.4	98.0	95.8
ambiguity tag / word		4.0	3.8	2.8	3.4	2.6

Table 7.6: Evaluation of the morphological analyzer

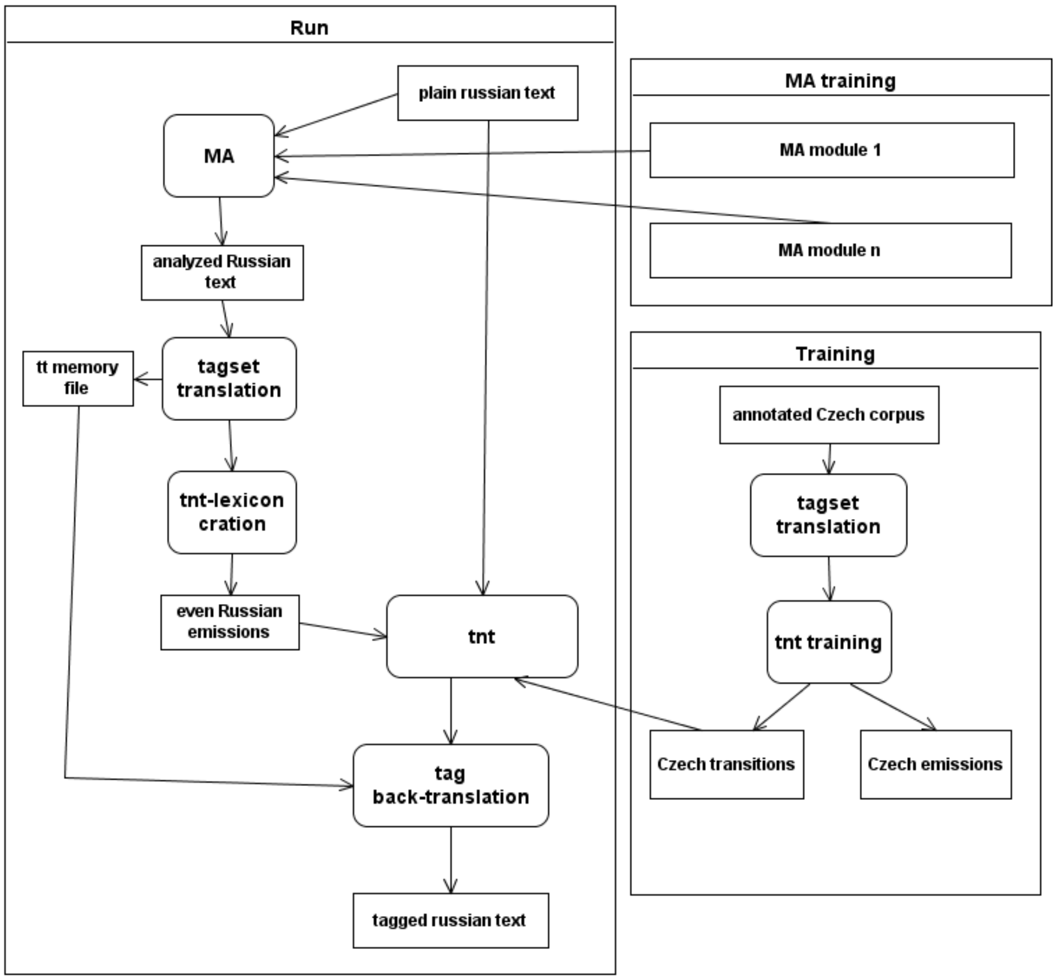

Figure 7.1: Schema of the Even tagger

Back to Experiment2: Using MA to approximate emissions

The *Direct* tagger (see §7.4) used Czech emissions to approximate Russian emissions. The previous section suggested that this is the main culprit of its poor performance. The Czech emissions can differ from the ideal Russian emissions in three ways:

1. A particular emitted word does not exist.
2. The set of tags associated with an emitted word is different.
3. The distribution of tags in that set is different.

The emissions almost certainly differ in all three ways. For example, as we mentioned in the evaluation of the *Direct* tagger, 55% of tokens in the Russian corpus did not occur in the Czech training corpus.

The results in Table 7.8 show that the accuracy clearly improved for all major open classes, especially, for verbs. The much lower accuracy for nouns (54.4%) and adjectives (53.1%) than for verbs (90.1%) is expected. In the output of the morphological analyzer that is the basis for the emissions, verbs have the ambiguity of 1.6 while the ambiguity for nouns and adjectives is 4.3 and 5.7, respectively (see the last column in §7.7). Moreover, verbs have also a higher recall.

7.6 Experiment 3: Approximating emissions with cognates

Although it is true that forms and distributions of Czech and Russian words are not the same, they are also not completely unrelated. As any Czech speaker would agree, the knowledge of Czech words *is* useful when trying to understand a text in Russian (obviously, one has to understand the script, as most Czechs do). The reason is that many of the corresponding Czech and Russian words are cognates, (i.e., historically they descend from the same ancestor root or they are mere translations).

Cognate pair

We define a *cognate pair* as a translation pair where words from two languages share both meaning and a similar surface form. Depending on how closely the two languages are related, they may share more or fewer cognate pairs. Linguistic intuition suggests that the information about cognate words in Czech should help in tagging Russian. Two hypotheses are tested in the experiments with respect to cognates:

1. Cognate pairs have similar morphological and distributional properties.
2. Cognate pairs are similar in form.

Obviously both of these assumptions are approximations because

1. Cognates could have departed in their meaning, and thus probably have different distributions. For example, consider *život* 'life' in Czech vs. *život* 'belly' in Russian, and *krásný* (adj.) 'nice' in Czech vs. *krasnyj* (adj.) 'red' in Russian.
2. Cognates could have departed in their morphological properties. For example, *tema* 'theme', borrowed from Greek, is neuter in Czech and feminine in Russian.
3. There are false cognates — unrelated, but similar or even identical words. For example, *dělo* 'cannon' in Czech vs. *delo* 'matter, affair' in Russian, *jel* [jɛl] 'drove' in Czech vs. *el* [jɛl] 'ate' in Russian, *pozor* 'attention' vs. *pozor* 'disgrace' in Russian, *ni* 'she$_{loc}$' in Czech vs. *ni* negative particle in Russian (corresponding to Czech *ani*).[1]

Nevertheless, the assumption here is that these examples are true exceptions from the rule and that in the majority of cases, cognates will look and behave similarly. The borrowings, counter-borrowings, and parallel developments of both Slavic and Romance languages have

[1]It is interesting that many unrelated languages have amazing coincidences. For example, the Russian *gora* 'mountail/hill' and the Konda *goro* 'mountain/hill' do not seem related; or the Czech *mlada* 'young' is a false cognate with the Arabic *malad* 'youth', but coincidentally, the words have similar meanings. This is definitely not a very frequent language phenomenon, but even though the words are not etymologically related, finding such pairs should not hurt the performance of the system.

been extensively studied (see e.g., Derksen 2008 for Slavic and Gess and Arteaga 2006 for Romance), but this book does not provide a survey of this research.

In Feldman et al. 2005, we report the results of an experiment where 200 most frequent nouns from the Russian development corpus are manually translated into Czech. They constitute about 60% of all noun tokens in the development corpus. The information about the distribution of the Czech translations is transferred into the Russian model using an algorithm similar to the one outlined in §7.6. The performance of the tagger that uses manual translations of these nouns improves by 10% on nouns and by 3.5% overall. The error analysis reveals that some Czech-Russian translations do not correspond well in their morphological properties and, therefore, create extra errors in the transfer process. However, overall the accuracy does improve.

Obviously, if we want to stay in the resource/knowledge-light paradigm, we cannot provide the list manually. The following section describes a language-independent algorithm for achieving comparable results.

Identifying cognates

Our approach to cognate detection does not assume access to philological erudition, to accurate Czech-Russian translations, or even to a sentence-aligned corpus. None of these resources would be obtainable in a resource-poor setting. Instead we simply look for similar words, using a modified edit distance (Levenshtein 1966) as a measure of similarity.

We use a variant of the edit distance where the cost of operations is dependent on the arguments. In general, we assume that characters sharing certain phonetic features are closer than characters not sharing them (we use spelling as an approximation of pronunciation – in both Russian and Czech the relation between spelling and pronunciation is relatively simple). Thus for example, *b* is closer to *p* than to, say, *j*. In addition, costs are refined based on some well-known and common language-specific phonetic-orthographic regularities. The non-standard distances for Czech and Russian include for example:

- Russian *è* and *e* have zero distance from Czech *e*.
- Czech *h* and *g* have zero distance from Russian *g* (in Czech, the original Slavic *g* was replaced by *h*, in Russian it was not).
- The length of Czech vowels is ignored (in Russian, vowel length is not phonemic)
- *y* and *i* are closer to each other than other vowels (modern Czech does not distinguish between them in pronunciation)

However, performing a detailed contrastive morpho-phonological analysis is undesirable, since portability to other languages is a crucial feature of the system. So, some facts from a simple grammar reference book should be enough. Ideally, optimal distances should be calculated; however, currently we set them based on our intuition.

To speed up the computation of distances we preprocess the corpora, replacing every character that has a unique zero-distance counterpart by that counterpart. At the end of the cognate acquisition processes, the cognates are translated back to their original spelling. Because edit distance is affected by the number of arguments (characters) it needs to consider, the edit distance measure is normalized by word length. The list of cognates includes all Czech-Russian pairs of words whose distance is a below certain threshold. We further require that the words have the same morphological features (except for the gender of nouns and the variant as they are lexical features).

Using cognates

The list of cognates obtained by the procedure described above is used to map the Czech emission probabilities to Russian emissions. To further explain this, assume w_{cze} and w_{rus} are cognate words. Let T_{cze} denote the tags that w_{cze} occurs with in the Czech training corpus. Let $p_{cze}(t)$ be the emission probability of tag t ($t \notin T_{cze} \Rightarrow p_{cze}(t) = 0$). Let T_{rus} denote tags assigned to w_{rus} by the morphological analyzer; $\frac{1}{|T_{rus}|}$ is the even emission probability. Then, assign the new emission probability $p'_{rus}(t)$ to every tag $t \in T_{rus}$ as is given in (23) (followed by normalization):

(23) $$p_{rus}{}'(t) = \begin{cases} p_{cze}(t) + \frac{1}{|T_{rus}|} & \text{if } t \in T_{rus} \\ 0 & \text{otherwise} \end{cases}$$

The results are presented in Table 7.9. For comparison, we also show the results of the *Direct* (see §7.4) and *Even* taggers (see §7.5). In comparison with the *Even* tagger, the accuracy of the *Cognates* tagger improves in all measures (with the exception of SubPOS of nouns and adjectives, where it gives the same accuracy).

7.7 Experiment 4: Approximating transitions

We have experimented with simple modifications of the source corpora with the goal to make their syntactic structure look more like the target language, thus resulting in transitions better approximating the transitions of the target language. This resulted in a modest improvement of tagging accuracy. However we do not plan to pursue this path any further in the near future because the required human effort is not justified by the results. For the languages we experimented with, the source transitions were fairly good approximations of target transitions (e.g., Czech transitions approximated very well Russian transitions).

7.8 Experiment 5: Voting

One of the problems when tagging with a large tagset is data sparsity; with 1,000 tags there are $1,000^3$ potential trigrams. It is very unlikely that a naturally occurring corpus will contain all the acceptable tag combinations with sufficient frequency to reliably distinguish them from the unacceptable combinations. However, not all morphological attributes are useful for predicting the attributes of the succeeding word (e.g., tense is not really useful for case).

In this section, we describe an experiment originally presented in (Hana et al. 2004). To overcome data sparsity issues, we trained a tagger on individual components of the full tag in the hope that the reduction of the tagset of each such sub-tagger reduces data sparsity. Unfortunately, the method did not improve the results as we had hoped. It does increase accuracy of the less effective taggers (e.g., *Even* from §7.5 or a similar tagger described in

Table 7.7: Evaluation of the Russian morphological analyzer

Lexicon		no	yes	no	yes
LEO		no	no	yes	yes
All	Recall error:	2.9	4.3	12.7	6.6
	ambiguity (tag/w)	9.7	4.4	3.3	2.8
N	Recall error:	2.6	4.9	41.6	13.7
	ambiguity (tag/w)	18.6	6.8	6.5	4.3
A	Recall error:	6.2	7.0	8.1	7.5
	ambiguity (tag/w)	21.6	10.8	3.3	5.7
V	Recall error:	0.8	2.0	2.3	2.3
	ambiguity (tag/w)	14.7	4.8	1.5	1.5

Table 7.8: Tagging with evenly distributed output of Russian MA

tagger name		Direct	Even
transitions		Czech	Czech
emissions		Czech	uniform Russian MA
All	Full tag:	48.1	77.6
	SubPOS	63.8	91.2
N	Full tag:	37.3	54.4
	SubPOS	81.1	89.6
A	Full tag:	31.7	53.1
	SubPOS	51.7	86.9
V	Full tag:	39.9	90.1
	SubPOS	48.1	95.7

Table 7.9: Tagging Russian using cognates

tagger name		Direct	Even	Cognates
transitions		Czech	Czech	Czech
emissions		Czech	even MA	cognates
All	Full tag:	48.1	77.6	79.5
	SubPOS	63.8	91.2	92.2
N	Full tag:	37.3	54.4	57.3
	SubPOS	81.1	89.6	89.9
A	Full tag:	31.7	53.1	54.5
	SubPOS	51.7	86.9	86.9
V	Full tag:	39.9	90.1	90.6
	SubPOS	48.1	95.7	96.1

the original paper), but not of those with higher accuracy. The results are still interesting for at least two reasons. First, it shows that a smaller tagset does not necessarily lead to an increase of accuracy. Second, it is possible, and even likely that it is possible to modify the basic method in a way that would indeed lead to improved results.

Tag decomposition

We focus on six positions — POS (p), SubPOS (s), gender (g), number (n), case (c), and person (e). The selection of the slots is based on linguistic intuition. For example, because a typical Slavic NP has the structure of (Det) A* N (NP$_{gen}$) PP* (very similar to English), it is reasonable to assume that the information about part of speech and agreement features (gender, number, case) of previous words should help in the prediction of the same slots of the current word. Likewise, information about part-of-speech, case and person should assist in determining person (finite verbs agree with the subject, subjects are usually in nominative). On the other hand, the combination of tense and case is *prima facie* unlikely to be much use for prediction. Indeed, most of the expectations are confirmed in the results.

Table 7.10: Russian tagger performance trained on individual slots vs. tagger performance trained on the full tag

	full tag	POS	SubPOS	gender	number	case
1 (POS)	92.2	92.0	–	–	–	–
2 (SubPOS)	91.3	–	90.1	–	–	–
3 (gender)	89.9	–	–	89.4	–	–
4 (number)	94.1	–	–	–	92.1	–
5 (case)	87.2	–	–	–	–	82.6

Table 7.11: Russian tagger performance trained on the combination of two features vs. tagger performance trained on the full tag

Feature 1	full tag	POS	gender	gender	number	case	case
Feature 2		case	case	negation	case	person	tense
1 (POS)	92.2	91.9	–	–	–	–	–
2 (SubPOS)	91.3	–	–	–	–	–	–
3 (gender)	89.9	–	89.7	89.2	–	–	–
4 (number)	94.1	–	–	–	93.2	–	–
5 (case)	87.2	85.6	85.6	–	84.7	82.9	83.3
8 (person)	99.2	–	–	–	–	98.9	–
9 (tense)	98.6	–	–	–	–	–	98.4
11 (negation)	96.0	–	–	*96.3	–	–	–

The performance of some of the models on the Russian development corpus is summarized in Tables 7.10, 7.11, and 7.12. All models are based on the *Russified* tagger (see §7.7), with the full-tag tagger being identical to it. The numbers marked by an asterisk indicate instances in which the sub-tagger outperforms the full-tag tagger. As can be seen, all the taggers trained on individual positions are worse than the full-tag tagger on those positions. This proves that a smaller tagset does not necessarily imply that tagging is easier (see Elworthy 1995, and Chapter 5 of (Feldman and Hana 2010)). Similarly, there is no improvement from the combination of unrelated slots — case and tense or gender and negation. However, combinations of (detailed) part-of-speech information with various agreement features (e.g.,

Table 7.12: Russian tagger performance trained on the combination of three or four features vs. tagger performance trained on the full tag

Feature 1	full tag	POS	POS	SubPOS	SubPOS	SubPOS
Feature 2		gender	number	gender	number	gender
Feature 3		case	case	case	case	number
Feature 4						case
1 (POS)	92.2	91.8	*92.3	*92.4	*92.5	*92.4
2 (SubPOS)	91.3	–	–	90.5	90.5	90.6
3 (gender)	89.9	89.6	–	89.6	–	*90.2
4 (number)	94.1	–	94.0	–	93.8	*94.3
5 (case)	87.2	86.3	*87.3	86.7	87.1	*87.6

SubPOS, number, and case) outperform the full-tag tagger on at least some of the slots. All of the improvements are quite modest.

Combining sub-taggers

The next step is to put the sub-tags back together to produce estimates of the correct full tags, and to see how performance is affected. Simply combining the values offered by the best taggers for each slot is not possible because that could yield illegal tags (e.g., nouns in past tense). Instead, we let the taggers choose the best tag from the tags offered by the morphological analyzer.

There are many possible formulas that could be used. We used the formula in (24):

(24) $\mathsf{bestTag} = \mathsf{argmax}_{t \in T_{MA}} \mathsf{val}(t)$

where:
1. T_{MA} is the set of tags offered by MA
2. $\mathsf{val}(t) = \sum^{14}{}_{k=0} N_k(t)/N_k$
3. $N_k(t)$ is the # of taggers voting for k-th slot of t
4. N_k is the total # of taggers on slot k

This formula means that the best tag is the tag that receives the highest average percentage of votes for each of its slots. Weighting slots is also possible in the val function if certain slots are more important than others; however we did not use this option.

We ran a number of possible sub-tagger combinations, using 1-4 taggers for each slot. Unfortunately, none of the resulting taggers outperformed the *Russified* tagger, the tagger they are based on, on the full tag (although some did on some of the individual slots). As an example, Table 7.13 reports the performance of a system where the three best taggers for a particular slot vote on that slot. The better accuracy for a given criterion is marked by an asterisk. The tagger is clearly worse than the original tagger on all tokens (77.2% vs. 80.0%).

Even though, intuitively, it seemed that the tagger decomposition approach should improve the overall performance of the system, our experiments have shown the opposite. One of the guesses that we can make here is that the tag decomposition was based on our linguistic intuition and it is unclear whether such an approach is the most optimal. We suggest to explore alternative tag decomposition techniques, such as the random decomposition used in error-correcting output coding (Dietterich and Bakiri 1991). This could shed interesting

Table 7.13: Voted classifier

		Russified (§7.7)	sample voting tagger
All	Full tag:	*80.0	77.2
	SubPOS	92.3	92.3
N	Full tag:	57.1	57.1
	SubPOS	89.3	*89.9
A	Full tag:	*55.9	53.8
	SubPOS	86.9	86.9
V	Full tag:	*92.7	82.8
	SubPOS	96.6	96.6

light on why the experiments described in this chapter were unsuccessful and how to further improve the tagging performance.

Chapter 8

Practical aspects

In this section we address the problem of collection, selection and creation of resources needed by the system described above.

8.1 Resources

The following resources must be **available**:

- a reference grammar book for information about paradigms and closed class words,
- a large amount of plain text for learning a lexicon, e.g. newspapers from the Internet,
- a large annotated training corpus of a related language,
- optionally, a dictionary (or a native speaker) to provide analyses of the most frequent words,
- a non-expert (not a linguist and not a native speaker) to create the resources listed below,
- limited access to a linguist (to make non-obvious decisions in the design of the resources),
- limited access to a native speaker (to annotate a development corpus, to answer a limited number of language specific questions).

and these resources must be **created**:

- a list of morphological paradigms,
- a list of closed class words with their analyses,
- optionally, a list of the most frequent forms,
- a small annotated development corpus.

For evaluation, an annotated test corpus must be also created. As this corpus is not part of the resource-light system per se, it can (and should) be as large as possible.

8.2 Restrictions

Since our goal is to create resources cheaply and fast, we intentionally limit (but not completely exclude) the inclusion of any linguist and of anybody knowing the target language. We also limit the time of training and encoding of the basic target-language linguistic information to a minimum.

8.3 Tagset

In traditional settings, a tagset is usually designed by a linguist, moreover a native speaker. The constraints of a resource-light system preclude both of these qualifications. Instead, we have standardized the process as much as possible to make it possible to have the tagset designed by a non-expert.

Positional Tagset

All languages we work with are morphologically rich. Naturally, such languages require a large number of tags to capture their morphological properties. An obvious way to make it manageable is to use a structured system. In such a system, a tag is a composition of tags each coming from a much smaller and simpler atomic tagset tagging a particular morpho-syntactic property (e.g. gender or tense). This system has many benefits, as has been discussed in the previous chapters, including the 1) relative easiness for a human annotator to remember individual positions rather than several thousands of atomic symbols; 2) systematic morphological description; 3) tag decomposability; and 4) systematic evaluation.

Tagset Design: Procedure

Instead of starting from scratch each time a tagset for a new language is created, we have provided an annotated tagset template. A particular tagset can deviate from this template, but only if there is a linguistic reason. The tagset template includes the following items:

- order of categories (POS, SubPOS, gender, animacy, number, case, ...) – not all might be present in that language; additional categories might be needed;
- values for each category (N – nouns, C – numerals, M – masculine);
- which categories we do not distinguish, even though we could (proper vs. common nouns);
- a fully worked out commented example (as mentioned above).

Such a template not only provides a general guidance, but also saves a lot of time, because many of rather arbitrary decisions involved in any tagset creation are done just once (e.g., symbols denoting basic POS categories, should numerals be included as separate POS, etc.). As stated, a tagset may deviate from such a template, but only if there is a specific reason for it.

8.4 Resources for the morphological analyzer

Our morphological analyzer relies on a small set of morphological paradigms and a list of closed class and/or most frequent words.

Morphological paradigms

For each target language, we create a list of morphological paradigms. We just encode basic facts about the target language morphology from a standard grammar textbook. On average, the basic morphology of highly inflected languages, such as Slavic languages, are captured in 70-80 paradigms. The choices on what to cover involve a balance between precision, coverage and effort.

A list of frequent forms

Entering a lexicon entry is very costly, both in terms of time and knowledge needed. While it is usually easy (for a native speaker) to assign a word to one of the major paradigm groups, it takes considerably more time to select the exact paradigm variant differing only in one or two forms (in fact, this may be even idiolect-dependent). For example, in Czech, it is easy to see that the word *atom* 'atom' does not decline according to the neuter paradigm *město* 'town', but it takes more time to decide to which of the hard masculine inanimate paradigms it belongs. On the other hand, entering possible analyses for individual word forms is usually very straightforward. Therefore, our system uses a list of manually provided analyses for the most common forms.

Note that the process of providing the list of forms is not completely manual – the correct analyses are selected from those suggested on the basis of the words' endings. This can be done relatively quickly by a native speaker or by a non-native speaker with the help of a basic grammar book and a dictionary.

8.5 Documentation

Since the main idea of the project is to create resources quickly for an arbitrarily selected fusional language, we cannot possibly create annotation and language encoding manuals for each language. So, we created a manual that explains the annotation and paradigm encoding procedure in general and describes the main attributes and possible values that a language consultant needs to consider when working on a specific language. The manual has five parts:

1. How to summarize the basic facts about the morphosyntax of a language;
2. How to create a tagset
3. How to encode morphosyntactic properties of the target language in paradigms;
4. How to create a list of closed class words.

5. Corpus annotation manual

The instructions are mostly language independent (with some bias toward Indo-European languages), but contain a lot of examples from languages we have processed so far. These include suggestions how to analyze personal pronouns, what to do with clitics or numerals.

8.6 Procedure

The resource creation procedure involves at least two people: a native speaker who can annotate a development corpus, and a non-native speaker who is responsible for the tagset design, morphological paradigms, and a list of closed class words or frequent forms. Below we describe our procedure in more detail.

Tagset and MA resources creation

We have realized that even though we do not need a native speaker, some understanding of at least basic morphological categories the language uses is helpful. So, based on our experience, it is better to hire a person who speaks (natively or not) a language with some features in common. For example, for Polish, somebody knowing Russian is ideal, but even somebody speaking German (it has genders and cases) is much better than a person speaking only English. In addition, a person who had created resources for one language performs much better on the next target language. Knowledge comes with practice.

The order of work is as follows:

1. The annotator is given basic training that usually includes the following: 1) brief explanation of the purpose of the project; 2) tagset design; 3) paradigm creation.
2. The annotator summarizes the basic facts about the morphosyntax of a language,
3. The first version of the tagset is created.
4. The list of paradigms and closed-class words is compiled. During this process, the tagset is further adjusted.

Corpus annotation

The annotators do not annotate from scratch. We first run our morphological analyzer on the selected corpus; the annotators then disambiguate the output. We have created a support tool (`http://ufal.mff.cuni.cz/~hana/law.html`) that displays the word to be annotated, its context, the lemma and possible tags suggested by the morphological analyzer. There is an option to insert a new lemma and a new tag if none of the suggested items is suitable. The tags are displayed together with their natural language translation.

Naturally, we cannot expect the tagging accuracy to be 100%. There are many factors that contribute to the performance of the model:

1. target language morphosyntactic complexity,
2. source-language–target-language proximity,
3. quality of the paradigms,
4. quality of the cognate pairs (that are used for approximating emissions),
5. time spent on language analysis,
6. expertise of language consultants,
7. supporting tools.

Bibliography

Anderson, Stephen R. (1993). "Wackernagel's Revenge: Clitics, Morphology, and the Syntax of Second Position". In: *Language* 69, pp. 68–98.

Beesley, K. and L. Karttunen (2003). *Finite State Morphology.* CSLI Publications. University of Chicago Press.

Bémova, Alena, Jan Hajic, Barbora Hladká, and Jarmila Panevová (1999). "Morphological and Syntactic Tagging of the Prague Dependency Treebank". In: *Proceedings of ATALA Workshop.* Paris, France, pp. 21–29. URL: `http://quest.ms.mff.cuni.cz/pdt/doc/pdt-atala.ps`.

Brants, Thorsten (2000). "TnT - A Statistical Part-of-Speech Tagger". In: *Proceedings of 6th Applied Natural Language Processing Conference and North American chapter of the Association for Computational Linguistics annual meeting (ANLP-NAACL)*, pp. 224–231.

Brill, Eric (1995). "Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging". In: *Computational Linguistics* 21.4, pp. 543–565.

— (1999). "A Closer Look at the Automatic Induction of Linguistic Knowledge". In: *Learning Language in Logic*, pp. 49–56.

Chen, Stanley F. and Joshua T. Goodman (1996). "An Empirical Study of Smoothing Techniques for Language Modeling". In: *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL).* Santa Cruz, CA, pp. 310–318.

Church, Kenneth W. (1988). "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text". In: *Proceedings of the 2nd Conference on Applied Natural Language Processing.* Austin, Texas, pp. 136–143.

Civit, Montserrat (2000). *Guía para la anotación morfológica del corpus CLiC-TALP (Versión 3).* Tech. rep. WP-00/06. Barcelona, Catalunya: X-Tract Working Paper. Centre de Llenguatge i Computació (CLiC).

Cloeren, Jan (1993). "Toward A Cross-Linguistic Tagset". In: *Workshop On Very Large Corpora: Academic And Industrial Perspectives.*

Cucerzan, Silviu and David Yarowsky (2002). "Bootstrapping a Multilingual Part-of-speech Tagger in One Person-day". In: *Proceedings of the 6th Conference on Natural Language Learning (CoNLL).* Taipei, Taiwan, pp. 132–138.

Cutler, A., J.A. Hawkins, and G. Gilligan (1985). "The suffixing preference: a processing explanation". In: *Linguistics* 23, pp. 723–758.

Cutting, Doug, Julian Kupiec, Jan Pedersen, and Penelope Sibun (1992). "A Practical Part-of-speech Tagger". In: *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP).* Trento, Italy: Association for Computational Linguistics, pp. 133–140.

Daelemans, W., J. Zavrel, and S. Berck (1996). "MBT: A Memory-based Part of Speech Tagger-Generator". In: *Proceedings of the Fourth Workshop on Very Large Corpora (VLC)*, pp. 14–27.

Daelemans, Walter, Antal van den Bosch, and Jakub Zavrel (1999). "Forgetting Exceptions is Harmful in Language Learning". In: *Machine Learning* 34, pp. 11–43.

Derksen, Rick (2008). *Etymological Dictionary of the Slavic Inherited Lexicon.* Leiden Indo-European Etymological Dictionary Series 4. Brill Press.

DeRose, Stephen J. (1988). "Grammatical Category Disambiguation by Statistical Optimization". In: *Computational Linguistics* 14.1, pp. 31–39.

Dietterich, T. G. and G. Bakiri (1991). "Error-correcting Output Codes: a General Method for Improving Multiclass Inductive Learning Programs". In: *Proceedings of the Ninth AAAI National Conference on Artificial Intelligence.* Ed. by T. L. Dean and K. McKeown. Menlo Park, CA: AAAI Press, pp. 572–577.

Elworthy, David (1995). "Tagset Design and Inflected Languages". In: *7th Conference of the European Chapter of the Association for Computational Linguistics (EACL), From Texts to Tags: Issues in Multilingual Language Analysis SIGDAT Workshop.* Dublin, pp. 1–10.

Erjavec, Tomaž (2004). "MTEXT-EAST Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora". In: *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC'04, ELRA.* Paris, France, pp. 1535–1538.

— (2009). "MULTEXT-East Morphosyntactic Specifications: Towards Version 4". In: *Proceedings of the MONDILEX Third Open Workshop.* Bratislava, Slovakia.

— (2010). "MULTEXT-East Version 4: Multilingual Morphosyntactic Specifications, Lexicons and Corpora". In: *Proceedings of the LREC 2010 Third Open Workshop.* Malta.

Feldman, Anna (2006). "Portable Language Technology: A Resource-light Approach to Morpho-syntactic Tagging". PhD thesis. The Ohio State University.

Feldman, Anna and Jirka Hana (2010). *A Resource-light Approach to Morpho-syntactic Tagging.* Ed. by Christian Mair, Charles F. Meyer, and Nelleke Oostdijk. Language and Computers 70. Amsterdam–New York: Rodopi Press.

Feldman, Anna, Jirka Hana, and Chris Brew (2005). "Buy One, Get One Free or What to Do When Your Linguistic Resources are Limited". In: *Proceedings of the Third International Seminar on Computer Treatment of Slavic and East-European Languages (Slovko).* Bratislava, Slovakia.

— (2006). “Experiments in Cross-Language Morphological Annotation Transfer”. In: *Proceedings of Computational Linguistics and Intelligent Text Processing, CICLing.* Lecture Notes in Computer Science. Mexico City, Mexico: Springer-Verlag, pp. 41–50.

Franks, Steven and Tracy Holloway King (2000). *A Handbook of Slavic Clitics.* Oxford University Press.

Fronek, Josef (1999). *English-Czech/Czech-English Dictionary.* Contains an overview of Czech grammar. Praha: Leda.

Gess, Randall Scott and Deborah L. Arteaga (2006). *Historical Romance Linguistics: Retrospective and Perspectives.* J. Benjamins.

Givón, Talmy (1979). *On Understanding Grammar.* New York: Academic Press.

Goldsmith, John (2010). “Segmentation and morphology”. In: *The Handbook of Computational Linguistics and Natural Language Processing.* Ed. by Chris Fox, Shalom Lappin, and Alexander Clark. Vol. 14. Wiley-Blackwell.

Greenberg, Joseph H. (1957). *Essays in Linguistics.* Chicago: University of Chicago Press.

Hajič, Jan (2004). *Disambiguation of Rich Inflection: Computational Morphology of Czech.* Prague, Czech Republic: Karolinum, Charles University Press.

Hajič, Jan and Barbora Hladká (1998). “Tagging Inflective Languages: Prediction of Morphological Categories for a Rich, Structured Tagset”. In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference (COLING-ACL).* Montreal, Canada, pp. 483–490.

Hall, Christopher J. (1988). “Integrating Diachronic and Processing Principles in Explaining the Suffixing Preference”. In: *Explaining Language Universals.* Ed. by J. A. Hawkins. Chap. 12.

Hana, Jiri (2007). “Czech Clitics in Higher Order Grammar”. PhD thesis. The Ohio State University.

Hana, Jiri, Anna Feldman, and Chris Brew (2004). “A Resource-light Approach to Russian Morphology: Tagging Russian using Czech resources”. In: *Proceedings of Empirical Methods for Natural Language Processing 2004 (EMNLP 2004).* Ed. by Dekang Lin and Dekai Wu. Barcelona, Spain: Association for Computational Linguistics, pp. 222–229. URL: `http://www.aclweb.org/anthology-new/W/W04/W04-3229.pdf`.

Hana, Jirka (2008). “Knowledge- and labor-light morphological analysis”. In: *OSUWPL* 58, pp. 52–84. URL: `http://ling.osu.edu/~hana/biblio/hana-2008-wp-morph.pdf`.

Hana, Jirka and Peter W. Culicover (2008). “Morphological Complexity Outside of Universal Grammar”. In: *OSUWPL* 58, pp. 85–109. URL: `http://ling.osu.edu/~hana/biblio/hana-culicover-2008.pdf`.

Hana, Jirka and Anna Feldman (2010). “A Positional Tagset for Russian”. In: *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010).* Valletta, Malta: European Language Resources Association, pp. 1278–1284. ISBN: 2-9517408-6-7.

Hana, Jirka, Anna Feldman, Luiz Amaral, and Chris Brew (2006). "Tagging Portuguese with a Spanish Tagger Using Cognates". In: *Proceedings of the Workshop on Cross-language Knowledge Induction hosted in conjunction with the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL).* Trento, Italy, pp. 33–40.

Haspelmath, Martin (2002). *Understanding Morphology.* Understanding Language. Arnold Publishers.

Hawkins, John A. and Gary Gilligan (1988). "Prefixing and suffixing universals in relation to basic word order". In: *Lingua* 74, pp. 219–259.

Hoeksema, Jack and Richard D. Janda (1988). "Implications of Process-morphology for Categorial Grammar". In: *Categorial Grammars and natural Language Structrues.* Ed. by Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler. Academic Press, pp. 199–247.

Ide, Nancy and Jean Véronis (1994). "MULTEXT-EAST: Multilingual Text Tools and Corpora". In: *Proceedings of the 15th International Conference on Computational Linguistics (COLING).* Vol. I. Kyoto, Japan, pp. 588–592.

Jelinek, Frederick (1985). "Markov Source Modeling of Text Generation". In: *Impact of Processing Techniques on Communication.* Ed. by F. K. Skwirzinski.

Johnson, Douglas C. (1972). *Formal Aspects of Phonological Description.* The Hague: Mouton.

Kaplan, Ronald M. and Martin Kay (1981). "Phonological rules and finite-state transducers". In: *Linguistic Society of America Meeting Handbook, Fifty-Sixth Annual Meeting.* New York.

Karlík, Petr, Marek Nekula, and Z. Rusínová (1996). *Příruční mluvnice češtiny [Concise Grammar of Czech].* Praha: Nakladatelství Lidové Noviny.

Karttunen, L. and K. Beesley (Saarijärvi, Finland). "Twenty-five years of finite-state morphology". In: *inquiries into Words, Constraints and Contexts (Festschrift in the Honour of Kimmo Koskenniemi and his 60th Birthday.* Gummerous Printing, pp. 71–83.

Karttunen, Lauri (1993). "Finite-state constraints". In: *The Last Phonological Rule.* Chicago, Illinois: University of Chicago Press.

Kazakov, Dimitar (1997). "Unsupervised learning of naïve morphology with genetic algorithms". In: *Workshop Notes of the ECML/MLnet Workshop on Empirical Learning of Natural Language Processing Tasks.* Ed. by W. Daelemans A. van den Bosch and A. Weijters. Prague, Czech Republic, pp. 105–112.

Klavans, Judith L. (1982). *Some problems in a theory of clitics.* Bloomington, IN: Indiana University Linguistics Club.

Kleene, S. C. (1956). "Representation of events in nerve nets and finite automata". In: *Automata Studies.* Ed. by C. E. Shannon and J. McCarthy. Princeton, NJ: Princeton University Press, pp. 3–41.

Koskenniemi, Kimmo (1983a). “Two-level Model for Morphological Analysis”. In: *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI).* Karlsruhe, Germany, pp. 683–685.

— (1983b). “Two-level morphology: a general computational model for word-form recognition and production”. PhD thesis. Helsinki: University of Helsinki.

— (1984). “A General Computational Model for Word-form Recognition and Production”. In: *Proceedings of the 10th International Conference on Computational Linguistics (COLING) and 22nd Annual Meeting of the Association for Computational Linguistics (ACL).* Stanford University, California, USA, pp. 178–181.

Levenshtein (1966). “Binary codes capable of correcting deletions, insertions, and reversals”. In: *Cybernetics and Control Theory* 10.8, pp. 707–710.

Marcken, Carl de (1995). “Acquiring a Lexicon from Unsegmented Speech”. In: *33rd Annual Meeting of the Association for Computational Linguistics (ACL).* Cambridge, Massachusetts, USA, pp. 311–313.

Marcus, Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz (1993). “Building a large annotated corpus of English: The Penn Treebank”. In: *Computational Linguistics* 19.2, pp. 313–330.

Merialdo, Bernard (1994). “Tagging English Text with a Probabilistic Model”. In: *Computational Linguistics* 20.2, pp. 155–171. ISSN: 0891-2017.

Mikheev, Andrei and Liubov Liubushkina (1995). “Russian Morphology: An Engineering Approach”. In: *Natural Language Engineering* 3.1, pp. 235–260.

Ratnaparkhi, Adwait (1996). “A Maximum Entropy Part-of-speech Tagger”. In: *Proceedings of the Empirical Methods in Natual Language Processing (EMNLP) Conference.* University of Pennsylvania, Philadelphia, USA, pp. 133–142.

Rijsbergen, C. J. van (1979). *Information Retrieval.* Butterworths, London.

Rissanen, Jorma (1989). *Stochastic Complexity in Statistical Inquiry.* Singapore: World Scientific Publishing Co.

Roark, Brian and Richard Sproat (2007). *Computational Approaches to Morphology and Syntax.* Oxford University Press.

Samuelsson, Christer (1993). “Morphological Tagging Based Entirely on Bayesian Inference”. In: *Proceedings of the 9th Nordic Conference on Computational Linguistics (NoDaLiDa).* Stockholm, Sweden.

Sapir, Edward (1921). *Language, an introduction to the study of speech.* New York: Harcourt.

Schmid, Helmut (1994). “Probabilistic Part-of-Speech Tagging Using Decision Trees”. In: *Proceedings of the International Conference on New Methods in Language Processing.* Manchester, UK.

Schone, P. and D. Jurafsky (2000). "Knowledge-Free Induction of Morphology Using Latent Semantic Analysis". In: *The 4th Conference on Computational Natural Language Learning and 2nd Learning Language in Logic Workshop.* Lisbon, Portugal, pp. 67–72.

Skoumalová, Hana (1997). "A Czech Morphological Lexicon". In: *Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology, Madrid.* ACL, pp. 41–47. URL: `http://arXiv.org/abs/cmp-lg/9707020`.

Viterbi, A.J. (1967). "Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm". In: *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Information Theory.* Vol. 13, pp. 260–269.

Weischedel, R., M. Meteer, R. Schwartz, L. Ramshaw, and J. Palmucci (1993). "Coping with Ambiguity and Unknown Words through Probabilistic Methods". In: *Computational Linguistics* 19.2, pp. 361–382.

Yarowsky, David and Richard Wicentowski (2000). "Minimally Supervised Morphological Analysis by Multimodal Alignment". In: *Proceedings of the 38th Meeting of the Association for Computational Linguistics (ACL)*, pp. 207–216.

Zipf, George K. (1935). *The Psychobiology of Language.* Houghton-Mifflin.

— (1949). *Human Behavior and the Principle of Least-Effort.* Addison-Wesley.

Zwicky, Arnold M. (1977). *On Clitics.* Tech. rep. Reproduced by the Indiana University Linguistics Club. Ohio State University.