1

Introduction: the Project

The project described in this book is at the very heart of linguistics; its goal is to describe, exhaustively and with absolute precision, all the sentences of a language likely to appear in written texts[1]. This project fulfills two needs: it provides linguists with tools to help them describe languages exhaustively (*linguistics*), and it aids in the building of software able to automatically process texts written in natural language (*natural language processing*, or NLP).

A linguistic project[2] needs to have a theoretical and methodological framework (how to describe this or that linguistic phenomenon; how to organize the different levels of description); formal tools (how to write each description); development tools to test and manage each description; and engineering tools to be used in sharing, accumulating, and maintaining large quantities of linguistic resources.

There are many potential applications of descriptive linguistics for NLP: spell-checkers, intelligent search engines, information extractors and annotators, automatic summary producers, automatic translators, etc. These applications have the potential for considerable economic usefulness, and it is therefore important for linguists to make use of these technologies and to be able to contribute to them.

1 Non-written languages, such as speech or sign language, can be transcribed by using specific alphabets, such as the International Phonetic Alphabet or the American Sign Alphabet. The resulting transcribed text indeed constitutes a written text.

2 [SAU 16, BLO 33] were among the first to attempt to rationalize the description of languages.

For now, we must reduce the overall linguistic project of describing *all* phenomena related to the use of language, to a much more modest project: here, we will confine ourselves to seeking to describe the set of all of the sentences that may be written or read in natural-language texts. The goal, then, is simply to design a system capable of distinguishing between the two sequences below:

a) *Joe is eating an apple*

b) *Joe eating apple is an*

Sequence (a) is a grammatical sentence, while sequence (b) is not.

This project constitutes the mandatory foundation for any more ambitious linguistic projects. Indeed it would be fruitless to attempt to formalize text styles (stylistics), the evolution of a language across the centuries (etymology), variations in a language according to social class (sociolinguistics), cognitive phenomena involved in the learning or understanding of a language (psycholinguistics), etc. without a model, even a rudimentary one, capable of characterizing sentences.

If the number of sentences were finite – that is, if there were a maximum number of sentences in a language – we would be able to list them all and arrange them in a database. To check whether an arbitrary sequence of words is a sentence, all we would have to do is consult this database: it is a sentence if it is in the database, and otherwise it is not. Unfortunately, there are an infinite number of sentences in a natural language. To convince ourselves of this, let us resort to a *redictio ad absurdum*: imagine for a moment that there are *n* sentences in English.

Based on this finite number *n* of initial sentences, we can construct a second set of sentences by putting the sequence *Lea thinks that*, for example, before each of the initial sentences:

Joe is sleeping → *Lea thinks that Joe is sleeping*

The party is over → *Lea thinks that the party is over*

Using this simple mechanism, we have just doubled the number of sentences, as shown in the figure below.

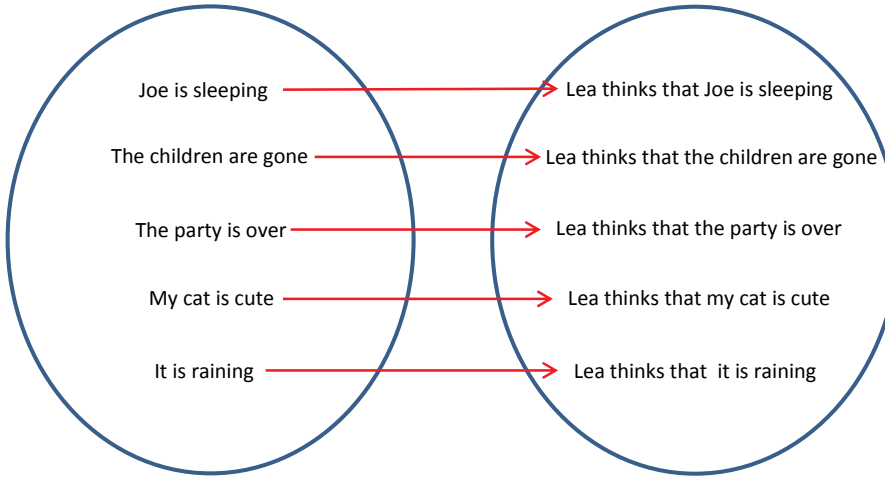

Figure 1.1. *The number of any set of sentences can be doubled*

This mechanism can be generalized by using verbs other than the verb *to think*; for example:

Lea (believes | claims | dreams | knows | realizes | thinks | ...) that Sentence.

There are several hundred verbs that could be used here. Likewise, we could replace *Lea* with several thousand human nouns:

(The CEO | The employee | The neighbor | The teacher | ...) thinks that Sentence.

Whatever the size n of an initial set of sentences, we can thus construct $n \times 100 \times 1{,}000$ sentences simply by inserting before each of the initial sentences, sequences such as *Lea thinks that*, *Their teacher claimed that*, *My neighbor declared that*, etc.

Language has other mechanisms that can be used to expand a set of sentences exponentially. For example, based on n initial sentences, we can construct $n \times n$ sentences by combining all of these sentences in pairs and inserting the word *and* between them. For example:

It is raining + *Joe is sleeping* → *It is raining and Joe is sleeping*

This mechanism can also be generalized by using several hundred connectors; for example:

It is raining (but | nevertheless | therefore | where | while |...) Joe is sleeping.

These two mechanisms (linking of sentences and use of connectors) can be used multiple times in a row, as in the following:

> *Lea claims that Joe hoped that Ida was sleeping.*
> *It was raining while Lea was sleeping, however Ida is now waiting, but the weather should clear up as soon as night falls.*

Thus these mechanisms are said to be *recursive*; the number of sentences that can be constructed with recursive mechanisms is infinite. Therefore it would be impossible to define all of these sentences *in extenso*. Another way must be found to characterize the set of sentences.

1.1. Characterizing a set of infinite size

Mathematicians have known for a long time how to define sets of infinite size. For example, the two rules below can be used to define the set of all natural numbers $\mathbb{N}$:

> (a) Each of the ten elements of set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} is a natural number;
>
> (b) any word that can be written as xy is a natural number if and only if its two constituents x and y are natural numbers.

These two rules constitute a formal definition of all natural numbers. They make it possible to distinguish natural numbers from any other object (decimal numbers or others). For example:

– Is the word “123” a natural number? Thanks to rule (a), we know that “1” and “2” are natural numbers. Rule (b) allows us to deduce from this that “12” is a natural number. Thanks to rule (a) we know that “3” is a natural number; since “12” and “3” are natural numbers, then rule (b) allows us to deduce that “123” is a natural number.

– The word “2.5” is not a natural number. Rule (a) enables us to deduce that “2” is a natural number, but it does not apply to the decimal point “.”. Rule (b) can only apply to two natural numbers, therefore it does not apply to the decimal point because it is not a natural number. In this case, “2.” is not a natural number; therefore “2.5” is not a natural number either.

There is an interesting similarity between this definition of set ℕ and the problem of characterizing the sentences in a language:

– Rule (a) describes *in extenso* the finite set of numerals that must be used to form valid natural numbers. This rule resembles a dictionary in which we would list all the words that make up the vocabulary of a language.

– Rule (b) explains how numerals can be combined to construct an infinite number of natural numbers. This rule is similar to grammatical rules that specify how to combine words in order to construct an infinite number of sentences.

To describe a natural language, then, we will proceed as follows: firstly we will define *in extenso* the finite number of basic units in a language (its vocabulary); and secondly, we will list the rules used to combine the vocabulary elements in order to construct sentences (its grammar).

1.2. Computers and linguistics

Computers are a vital tool for this linguistic project, for at least four reasons:

– From a theoretical point of view, a computer is a device that can verify automatically that an element is part of a mathematically-defined set. Our goal is then to construct a device that can automatically verify whether a sequence of words is a valid sentence in a language.

– From a methodological point of view, the computer will impose a framework to describe linguistic objects (words, for example) as well as the rules for use of these objects (such as syntactic rules). The way in which linguistic phenomena are described must be consistent with the system: any inconsistency in a description will inevitably produce an error (or “bug”).

– When linguistic descriptions have been entered into a computer, a computer can apply them to very large texts in order to extract from these texts examples or counterexamples that validate (or not) these descriptions. Thus a computer can be used as a scientific instrument (this is the *corpus linguistics* approach), as the telescope is in astronomy or the microscope in biology.

– Describing a language requires a great deal of descriptive work; software is used to help with the development of databases containing numerous linguistic objects as well as numerous grammar rules, much like engineers use computer-aided design (CAD) software to design cars, electronic circuits, etc. from libraries of components.

Finally, the description of certain linguistic phenomena makes it possible to construct NLP software applications. For example, if we have a complete list of the words in a language, we can build a spell-checker; if we have a list of rules of conjugation we can build an automatic conjugator. A list of morphological and phonological rules also makes it possible to suggest spelling corrections when the computer has detected errors, while a list of simple and compound terms can be used to build an automatic indexer. If we have bilingual dictionaries and grammars we can build an automatic translator, and so forth. Thus the computer has become an essential tool in linguistics, so much so that opposing "computational linguists" with "pure linguists" no longer makes sense.

1.3. Levels of formalization

When we characterize a phenomenon using mathematical rules, we *formalize* it. The formalization of a linguistic phenomenon consists of describing it, by storing both linguistic objects and rules in a computer. Languages are complicated to describe, partly because interactions between their phonological and writing systems have multiplied the number of objects to process, as well as the number of levels of combination rules. We can distinguish five fundamental levels of linguistic phenomena; each of these levels corresponds to a level of formalization.

> To analyze a written text, we access letters of the alphabet rather than words; thus it is necessary to describe the link between the alphabet and the orthographic forms we wish to process (*spelling*). Next, we must establish a link between the

orthographic forms and the corresponding vocabulary elements (*morphology*). Vocabulary elements are generally listed and described in a lexicon that must also show all potential ambiguities (*lexicography*). Vocabulary elements combine to build larger units such as phrases which then combine to form sentences; therefore rules of combination must be established (*syntax*). Finally, links between elements of meaning which form a predicate transcribed into an elementary sentence, as well as links between predicates in a complex sentence, must be established (semantics).

1.4. Not applicable

We do not always use language to represent and communicate information directly and simply; sometimes we play with language to create sonorous effects (for example in poetry). Sometimes we play with words, or leave some "obvious" information implicit because it stems from the culture shared by the speakers (anaphora). Sometimes we express one idea in order to suggest another (metaphor). Sometimes we use language to communicate statements about the real world or in scientific spheres, and sometimes we even say the opposite of what we really mean (irony).

It is important to clearly distinguish problems that can be solved within a strictly linguistic analytical framework from those that require access to information from other spheres in order to be solved.

1.4.1. *Poetry and plays on words*

Writers, poets, and authors of word games often take the liberty of constructing texts that violate the syntactic or semantic constraints of language. For example, consider the following text[3]:

For her this rhyme is penned, *whose luminous eyes*
Brightly expressive as the twins of Leda,
Shall find her own sweet name, *that nesting lies*,
Upon the page, *enwrapped from every reader.*

3 From the poem *A Valentine* by Edgar Allan Poe.

This poem is an acrostic, meaning that it contains a puzzle which readers are invited to solve. We cannot rely on linguistic analysis to solve this puzzle. But, to even understand that the poem is a puzzle, the reader must figure out that *this rhyme* refers to the poem itself. Linguistic analysis is not intended to figure out what in the world *this rhyme* might be referring to; much less to decide among the possible candidates.

… luminous eyes brightly expressive as the twins of Leda …

The association between the adjective *luminous* and *eyes* is not a standard semantic relationship; unless the eyes belong to a robot, eyes are not luminous. This association is, of course, metaphorical: we have to understand that *luminous eyes* means that the owner of the eyes has a luminous intelligence, and that we are perceiving this luminous intelligence by looking at her eyes.

The *twins of Leda* are probably the mythological heroes Castor and Pollux (the twin sons of Leda, the wife of the king of Sparta), but they are not particularly known for being *expressive*. These two heroes gave their names to the constellation *Gemini*, but I confess that I do not understand what *an expressive constellation* might be. I suspect the author rather meant to write:

… expressive eyes brightly luminous as the twins of Leda …

The associations between the noun *name* and the verbal forms *lies*, *nestling*, and *enwrapped* are no more direct; we need to understand that it is the written form of the name which is present on the physical page where the poem is written, and that it is hidden from the reader.

If we wish to make a poetic analysis of this text, the first thing to do is thus to note these non-standard associations, so we will know where to run each poetic interpretive analysis. But if we do not even know that *eyes* are not supposed to be *luminous*, we will not be able to even figure out that there is a metaphor, therefore we will not be able to solve it (i.e. to compute that the woman in question is intelligent), and so we will have missed an important piece of information in the poem. More generally, in order to understand a poem's meaning, we must first note the semantic violations it contains. To do this, we need a linguistic model capable of distinguishing

"standard" associations such as *an intelligent woman*, *a bright constellation*, *a name written on a page*, etc. from associations requiring poetic analysis, such as *luminous eyes*, *an expressive constellation*, *a name lying upon a page*.

Analyzing poems can pose other difficulties, particularly at the lexical and syntactic levels. In standard English, word order is less flexible than in poems. To understand the meaning of this poem, a modern reader has to start by rewriting (in his or her mind) the text in standard English, for example as follows:

> *This rhyme is written for her, whose luminous eyes (as brightly expressive as the twins of Leda) will find her own sweet name, which lies on the page, nestling, enwrapped from every reader.*

> The objective of the project described in this book is to formalize standard language without solving poetic puzzles, or figuring out possible referents, or analyzing semantically non-standard associations.

1.4.2. *Stylistics and rhetoric*

Stylistics studies ways of formulating sentences in speech. For example, in a text we study the use of understatements, metaphors, and metonymy ("figures of style"), the order of the components of a sentence and that of the sentences in a speech, and the use of anaphora. Here are a few examples of stylistic phenomena that cannot be processed in a strictly linguistic context:

Understatement: *Joe was not the fastest runner in the race*

Metaphor: *The CEO is a real elephant*

Metonymy: *The entire table burst into laughter*

In reality, the sentence *Joe was not the fastest runner in the race* could mean here that Joe came in last; so, in a way, this sentence is not saying what it is expressing! Unless we know the result of the race, or have access to information about the real Joe, we cannot expect a purely linguistic analysis system to detect understatements, irony or lies.

To understand the meaning of the sentence *The CEO is a real elephant*, we need to know firstly that a CEO cannot really be an elephant, and therefore that this is a metaphor. Next we need to figure out which "characteristic property" of elephants is being used in the metaphor. Elephants are known for several things: they are big, strong, and clumsy; they have long memories; they are afraid of mice; they are an endangered species; they have big ears; they love to take mud-baths; they live in Africa or India, etc. Is the CEO clumsy? Is he/she afraid of mice? Does he/she love mud-baths? Does he/she have a good memory? To understand this statement, we would have to know the context in which the sentence was said, and we might also need to know more about the CEO in question.

To understand the meaning of the sentence *The entire table burst into laughter*, it is necessary first to know that a table is not really capable of bursting into laughter, and then to infer that there are people gathered around a table (during a meal or a work meeting) and that it is these people who burst out laughing. The noun *table* is neither a collective human noun (such as *group* or *colony*), nor a place that typically contains humans (such as *meeting room* or *restaurant*), nor an organization (such as *association* or *bank*); therefore using only the basic lexical properties associated with the noun *table* will not be enough to comprehend the sentence.

It is quite reasonable to expect a linguistic system to detect that the sentences *The CEO is a real elephant* and *The entire table burst into laugther* are not standard sentences; for example, by describing *CEO* as a human noun, describing *table* as a concrete noun, and requiring *to burst into laughter* to have a human subject, we can learn from a linguistic analysis that these sentences are not "standard", and that it is therefore necessary to initiate an extra-linguistic computation such as metaphor or metonymy calculations in order to interpret them.

> The linguistic project described in this book is not intended to solve understatements, metaphors, or metonymy, but it must be able to detect sentences that are deviant in comparison to the standard language.

1.4.3. *Anaphora*, *coreference resolution*, *and semantic disambiguation*

> Coreference*: Lea invited Ida for dinner. She brought a bottle of wine.*

Anaphora: *Phelps returned. The champion brought back 6 medals with him.*

Semantic ambiguity: *The round table is in room B17.*

In order to understand that in the sentence *She brought a bottle of wine*, *she* refers to *Ida* and not *Lea*, we need to know that it is usually the guest who travels and brings a bottle of wine. This social convention is commonplace throughout the modern Western world, but we would need to be sure that this story does not take place in a society where it is the person who invites who brings beverages.

In order to understand that *The champion* is a reference to Phelps, we have to know that Phelps is a champion. Note that dozens of other nouns could have been used in this anaphora: *the American*, *the medal-winner*, *the record-holder*, *the swimming superstar*, *the young man*, *the swimmer*, *the former University of Florida student*, *the breakaway*, *the philanthropist*, etc.

In order to eliminate the ambiguity of the sequence *round table* (between "a table with a round shape" and "a meeting"), we would need to have access to a wider context than the sentence alone.

> The linguistic project described in this book is not intended to resolve anaphora or semantic ambiguities.

NOTE. – I am not saying that it is impossible to process poetry, word games, understatements, metaphors, metonymy, coreference, anaphora, and semantic ambiguities; I am only saying that these phenomena lie outside the narrow context of the project presented in this book. There are certainly "lucky" cases in which linguistic software can automatically solve some of these phenomena. For example, in the following sequence:

Joe invited Lea for dinner. She brought a bottle of wine

a simple verification of the pronoun's gender would enable us to connect *She* to *Lea*. Conversely, it is easy to build software which, based on the two sentences *Joe invited Lea to dinner* and *Lea brought a bottle of wine*, would produce the sentence *She brought a bottle of wine*. Likewise, in the sentence:

The round table is taking place in room B17

a linguistic parser could automatically figure out that the noun *round table* refers to a meeting, provided that it has access to a dictionary in which the noun *round table* is described as being an abstract noun (synonymous with *meeting*), and the verb *to take place* is described as calling for an abstract subject.

1.4.4. *Extralinguistic calculations*

Consider the following statements:

a) *Two dollars plus three dollars make four dollars.*

b) *Clinton was already president in 1536.*

c) *The word God has four letters.*

d) *This sentence is false.*

These statements are expressed using sentences that are well-formed because they comply with the spelling, morphological, syntactic, and semantic rules of the English language. However, they express statements that are incorrect in terms of mathematics (a), history (b), spelling (c), or logic (d). To detect these errors we would need to access knowledge that is not part of our strictly linguistic project[4].

> The project described in this book is confined to the formalization of language, without taking into account speakers' knowledge about the real world.

1.5. NLP applications

Of course, there are fantastic software applications capable of processing extralinguistic problems! For example, the IBM computer Watson won on the game show *Jeopardy!* in spectacular fashion in 2011; I have a lot of fun

4 Some linguists have put forward meta-linguistics examples of type (c) to demonstrate the necessity of using unrestricted grammar to describe languages. In addition, many NLP researchers use phenomena such as metaphors, resolution of anaphora, detection of understatement, etc. to demonstrate the inadequacy of linguistic techniques to handle… what I consider extralinguistic phenomena.

asking my smart watch questions. In the car, I regularly ask Google Maps to guide me verbally to my destination; my language-professor colleagues have trouble keeping their students from using Google Translate; and the subtitles added automatically to YouTube videos are a precious resource for people who are hard of hearing [GRE 11], etc.

All of these software platforms have a NLP part, which analyzes or produces a written or verbal statement, often accompanied by a specialized module, for example a search engine or GPS navigation software. It is important to distinguish between these components: just because we are impressed by the fact that Google Maps gives us reliable directions, it does not mean it speaks perfect English. It is very possible that IBM Watson can answer a question correctly without having really "understood" the question. Likewise, a software platform might automatically summarize a text using simple techniques to filter out words, phrases or sentences it judges to be unimportant [MAN 01][5]. Word-recognition systems use signal processing techniques to produce a sequence of phonemes and then determine the most "probable" corresponding sequence of words by comparing it to a reference database [JUR 00][6], etc.

Most pieces of NLP software actually produce spectacular, almost magical results, with a very low degree of linguistic competence. To produce these results, the software uses "tricks", often based on statistical methods [MAN 99].

Unfortunately, the success of these software platforms is often used in order to show that statistical techniques have made linguistics unnecessary[7]. It is important, then, to understand their limitations. In the next two sections I will take a closer look at the performances of the two "flagship" statistical NLP software platforms: automatic translation and part-of-speech tagging.

5 [MAN 01] introduces the field of automatic summarizing and its issues, in particular how to detect the "important" information that should be included in a summary.

6 For the exception that proves the rule, see [RAY 06].

7 For an amusing read about the ravages of statistics in fields other than linguistics, see [SMI 14].

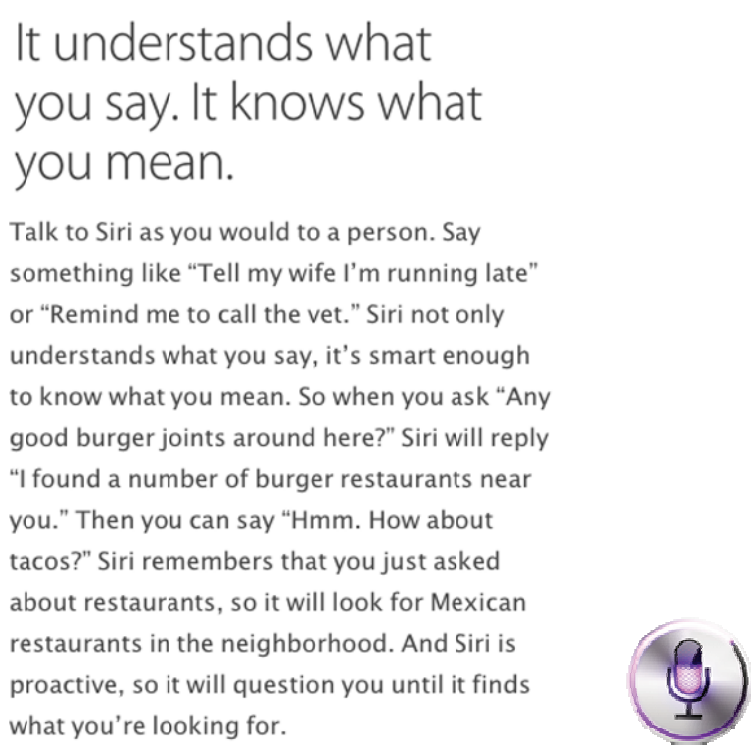

Figure 1.2. *Really?*

1.5.1. *Automatic translation*

Today, the best-known translation software platforms use statistical techniques to suggest a translation of texts. These software platforms are regularly cited as examples of the success of statistical techniques, and everyone has seen a "magical" translation demo[8]. It is not surprising, therefore, that most people think the problem of translation has already been solved. I do not agree.

8 For example, here is a demonstration of the instantaneous translation used by Google Glasses: www.youtube.com/watch?v=pZKWW3rzT2Q.

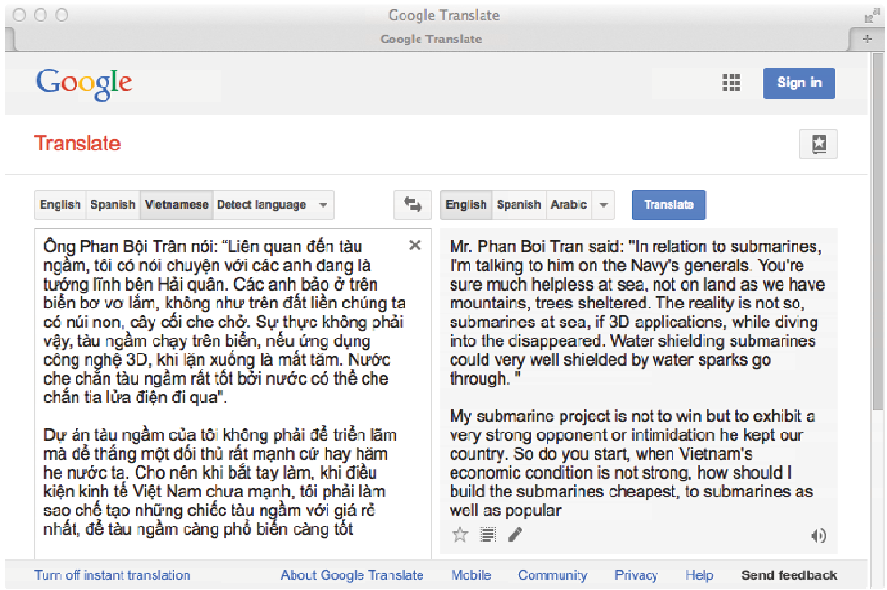

Figure 1.3. *Vietnamese–English translation with Google Translate*

For example, Figure 1.3 shows how Google Translate has translated part of an article in Vietnamese (www.thanhnieunnews.com, October 2014). The text produced does not make very much sense; what does "I'm talking to him on the Navy's generals" mean? The translated text even contains incorrect constructions ("could very well shielded", for example).

Figure 1.4 allows us to compare Google Translate's results with those obtained using a simple Arabic–English dictionary and a very basic translation grammar; see [BAN 15]. For example, the first sentence was wrongly translated by Google Translate as *The man who went down to work* instead of *The man who fell went to work*. Google Translate was also wrong about the second sentence, which means *The man that you knew went to work* and not *I knew a man who went to work*.

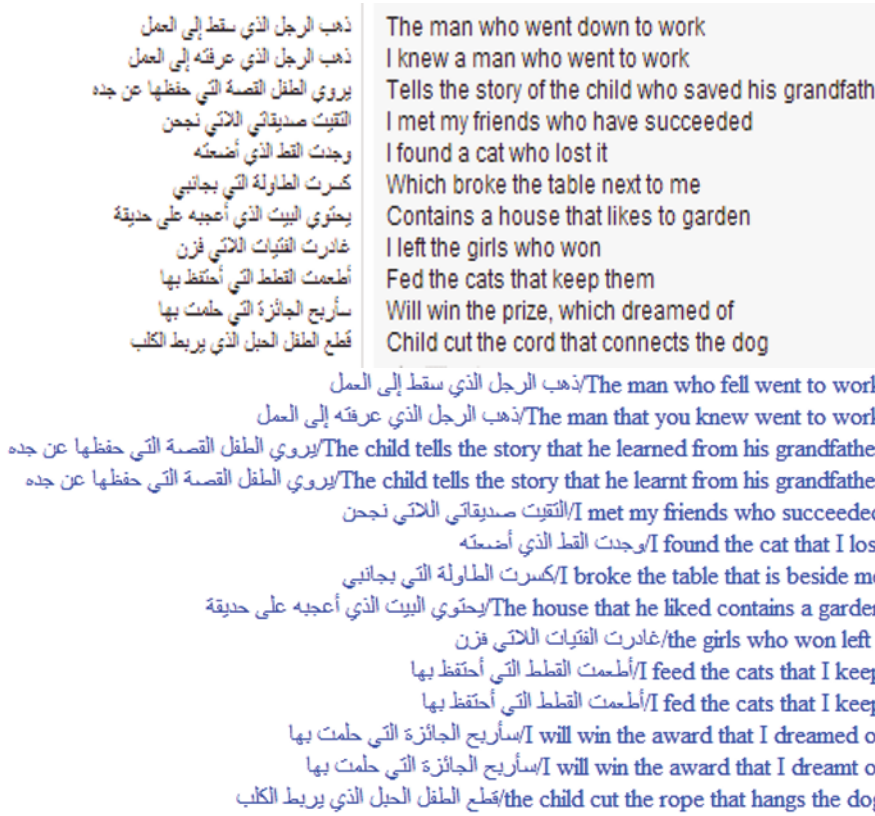

Figure 1.4. *Translation with Google Translate vs. with NooJ*

When translating languages that are more similar, such as French into English, the results produced by Google Translate are helpful, but still could not be used in a professional context or to translate a novel, a technical report, or even a simple letter, and especially not when submitting a resumé.

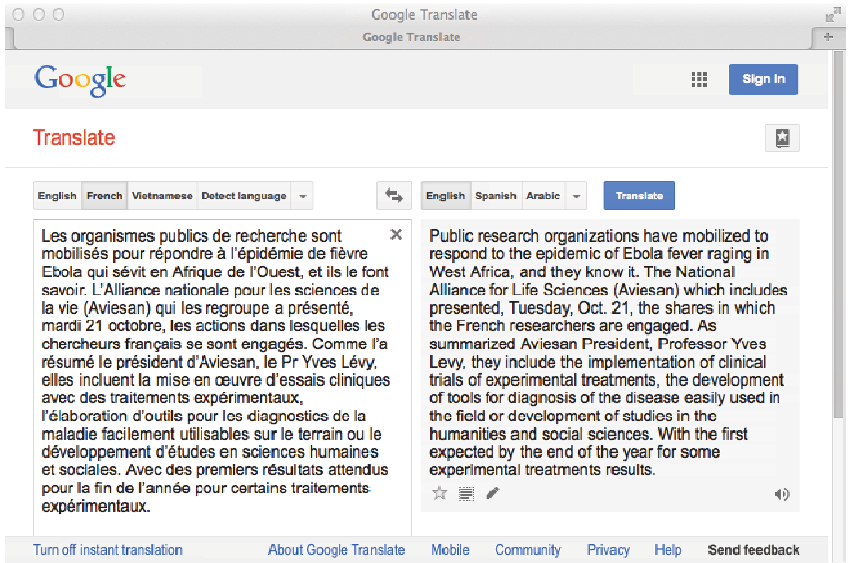

Figure 1.5. *Article from the newspaper Le Monde (October 2014) translated with Google Translate*

Let us look in detail at the result produced by Google Translate. None of the English sentences produced is correct:

– The first sentence has an opposite meaning; the expression *ils le font savoir* has been translated as *they know it* instead of *they make it known*.

– The second sentence has an ungrammatical sequence *...which includes presented...* The term *action* has been wrongly translated as *share* instead of *action*.

– In the third sentence, the verb *summarized* is placed incorrectly; *essais cliniques avec des traitements experimentaux* should be translated as *clinical trials* with *experimental treatments*; and *utilizable* should be translated as *useable* or *useful* and not as *used*.

– The term *results* is placed incorrectly in the last sentence.

To be fair, it should be noted that every attempt to construct good-quality automatic translation software has failed, including those based on linguistic techniques, such as the European program Eurotra (1978–1992). It is my belief that the reasons for Eurotra's failure have to do with certain scientific

and technical choices (as well as real problems with management), and not with a theoretical impossibility of using linguistics to do translations.

I will turn now to another flagship application, which is less "public-at-large" than machine translation, but just as spectacular for NLP specialists: part-of-speech tagging.

1.5.2. *Part-of-speech (POS) tagging*

Part-of-speech (POS) tagging is often presented as the basic application of any piece of NLP software, and has historically justified the sidelining of linguistic methods in favor of stochastic ones. The authors of tagging software frequently speak of 95% precision; these results seem "magical" too, since POS taggers use neither dictionaries nor grammars to analyze the words of any text with such a great precision. Linguists have difficulty justifying their painstaking descriptive work when shown what a computer can do by itself and without linguistic data! It is also commonplace to hear that taggers' results prove that statistical techniques have bypassed linguistic ones; for example:

> *Automatic part of speech tagging is an area of natural language processing where statistical techniques have been more successful than rule-based methods. [BRI 92]*

In their course on NLP (available on YouTube as of December 2015), Dan Jurafsky and Chris Manning consider the problem of the construction of POS taggers as "mostly solved"; more generally, NLP researchers use the spectacular results produced by statistical taggers to validate the massive use of statistical techniques in all NLP applications, always to the detriment of the linguistic approach.

A POS tagger is an automatic program that links each word in a text with a "tag", in practice its POS category: noun, verb, adjective, etc. To do this, taggers use reference corpora which have been manually tagged[9]. To analyze a

9 Some people use terms such as "training data" or "machine learning" to indicate that the tagger "learns" something when it "trains" on data. This fancy terminology is impressive, but in fact what POS taggers do is nothing more than copying and pasting codes from a reference text to the text they analyze.

text, the tagger examines the context of each word in the text and compares it with the contexts of the occurrences of this same word in the reference corpus in order to deduce which tag should be linked with the word.

Figure 1.6 shows, for example, an extract from Penn Treebank, which is one of the reference corpora[10] used by English POS taggers.

```
Battle-tested/SingularProperName Japanese/SingularProperName
industrial/Adjective managers/PluralNoun here/Adverb always/Adverb
buck/Verb up/Preposition nervous/Adjective newcomers/PluralNoun
with/Preposition the/Determiner tale/SingularNoun of/Preposition
the/Determiner first/Adjective of/Preposition their/PossessivePronoun
countrymen/PluralNoun to/TO visit/Verb Mexico/SingularProperName ,/,
a/Determiner boatload/SingularNoun of/Preposition samurai/PluralNoun
warriors/PluralNoun blown/PastParticiple ashore/Adverb 375/Number
years/PluralNoun ago/Adverb ./.
From/Preposition the/Determiner beginning/SingularNoun ,/,
it/PersonalPronoun took/PreteritVerb a/Determiner man/SingularNoun
with/Preposition extraordinary/Adjective qualities/PluralNoun to/TO
succeed/Verb in/Preposition Mexico/SingularProperName ,/, "/"
says/Present3rdSingularVerb Kimihide/SingularProperName
Takimura/SingularProperName ,/, president/SingularNoun of/Preposition
Mitsui/PluralNoun group/SingularNoun 's/Possessive
Kensetsu/SingularProperName Engineering/SingularProperName
Inc./SingularProperName unit/SingularNoun ./.
```

Figure 1.6. *Extract from Penn Treebank*

I do not believe that taggers should be considered as successes, and here are my reasons why.

1.5.2.1. *The results of statistical methods are not actually so spectacular*

The number of unambiguous words is so large compared to the very small number of tags used by taggers that a simple software application that

10 This extract is described in [MAR 93]. I have replaced the original codes (for example "JJ") with clearer codes ("Adjective").

would tag words just by copying the most frequent tag in the reference corpus would already have a degree of precision greater than 90% [CHA 97].

For example, in English the words *my*, *his*, *the* (always determiners), *at*, *from*, *of*, *with* (always prepositions), *him*, *himself*, *it*, *me*, *she*, *them*, *you* (always pronouns), *and*, *or* (always conjunctions), *again*, *always*, *not*, *rather*, *too* (always adverbs), *am*, *be*, *do*, *have* (always verbs), and *day*, *life*, *moment*, *thing* (always nouns) are extremely frequent but have only a single possible tag, and thus are always tagged correctly.

The vast majority of ambiguous words are actually favored in terms of analysis; for example, in most of their occurrences, the forms *age*, *band*, *card*, *detail*, *eye*, etc. represent nouns and not the verbs *to age*, *to band*, *to card*, *to detail*, *to eye*, etc. A software platform systematically disregarding the rare verbal hypothesis for these words will therefore almost never be wrong.

In these conditions, obtaining a 95% correct result when a simple copy already yields 90% precision is not really spectacular; on average we get one correct result out of two for difficult cases, which is more like a coin-toss than a feat of "learning".

> The degree of precision claimed by taggers is, in reality, not that impressive.

1.5.2.2. *POS taggers disregard the existence of multiword units*

Taggers do not take into account multiword units or expressions, though these frequently occur in texts[11]. In the Penn Treebank extract shown in Figure 1.6, the compound noun *industrial managers*, the phrasal verb *to buck up*, the compound determiner *a boatload of*, the compound noun *samurai warrior*, the expression *to blow N ashore*, the adverb *from the beginning*, and the expression *it takes N to V-inf* have all simply been disregarded.

However, processing *industrial manager* as a sequence of two linguistic units does not make any sense: an *industrial manager* is not a *manager* who

11 [SIL 95] shows that up to one-third of the forms present in the texts of the newspaper *Le Monde* are in fact composed of multiword units or expressions.

has been bred or made with industrial methods. Yet this analysis is the general analysis for the adjective *industrial*, for example as in *industrial diamond*, *industrial food*, *industrial soap*, *industrial chicken*, etc. Likewise, the phrasal verb *to buck up* (which means *to encourage*) should not be analyzed word-for-word as *to buck* (which means *to oppose*) followed by the locative preposition *up*. The head word of the noun phrase *a boatload of samurai warriors* is *warriors*, therefore *a boatload of* should be treated as a determiner, and above all *boatload* should not be treated as the head word of the noun phrase.

More generally, systematically tagging texts without taking into account multiword units, phrasal verbs and expressions eliminates any possibility of conducting meaningful linguistic analyses on the resulting tagged text. For example, tagging the sequence **blue**/Adjective **collar**/Noun even when it means *manual laborer* would block the analysis of sentences such as *The blue collars finally accepted their offer*, since the verb *to accept* expects a human subject, and not a part of a shirt. Likewise, disregarding the adverb *all of a sudden* by tagging it as **all**/Pronoun **of**/Preposition **a**/Determiner **sudden**/Adjective would make a syntactic parser crash, since this sequence of categories does not exist in English grammar.

The same is true for expressions, which are extremely numerous in vocabulary and texts: taggers simply disregard them. It is typical to see texts labeled as follows:

> **The**/Determiner **CEO**/Noun **took**/Verb **our**/Determiner **request**/Noun **into**/Preposition **account**/Noun

in which the expression *to take ... into account* has not been represented. It is hard to imagine how this tagged text will be processed, or even translated, if the computer really thinks that it is dealing with the verb *to take* (as in *Joe took his pen*) and the noun *account* (as in *an email account*).

I argue in this book that systematically disregarding multiword units and expressions inevitably leads to the production of incorrect analyses that are useless for most NLP applications, including search engines and automatic translation; imagine the quality of a translation in which the French

multiword units *tout de suite* [right now] or *carte bleue* [visa card] were translated word for word:

> *Je cherche ma carte bleue tout de suite* → * *I look for my blue card all of rest*

In practice, if multiword units and expressions were taken into account in order to assess the precision of POS taggers, that precision would fall below 70%, meaning that it would be inferior to the precision of a simple program that would access a dictionary that contains all of the multiword units and expressions in a language.

> By not processing frequent multiword units and expressions even though they constitute a non-negligible subset of vocabulary, taggers produce useless results.

1.5.2.3. *Statistical methods are costly*

We often hear that linguistic methods are costly to implement because they require the construction of dictionaries and grammars. However, statistical methods also require a great deal of work to manually construct their reference corpora. In the end, tagging a corpus is necessarily much more labor-intensive than constructing the equivalent dictionary, since a given word occurs multiple times in a corpus but only once in a dictionary, and it would be necessary to manually tag an extremely large corpus if we wished to cover only the standard vocabulary of a language (i.e. to construct a reference corpus in which each use of each standard word occurred at least once).

> The construction of the reference corpus required by taggers is an expensive operation.

1.5.2.4. *Reference corpora are not reliable*

One answer to the argument above is that the people who manually tag a reference corpus do not possess a level of qualification comparable to those who construct dictionaries and grammars, and can therefore be hired at a lower cost. In practice, the consequence of this cynical attitude is that most so-called "reference" corpora contain a high number of errors. For example, note the mistakes in the Penn Treebank extract shown in Figure 1.6: *Battle-tested* and *Japanese* should have been tagged as adjectives (and not proper nouns); *up* is a particle (and not a preposition), *Mitsui* is a proper name (not

a plural common noun); *Engineering* is a common noun; and *Inc*. is an adjective. A corpus considered as the "reference" by just about the entire NLP community in fact contains a large number of errors.

The fact that "reference corpora" contain errors is well known to NLP researchers, to the extent that the study of errors in reference corpora has become a sub-field of NLP research[12]. If reference corpora contain so many errors, we cannot expect the programs that use them to provide usable results.

> The reference corpora used by taggers are not reliable.

1.5.2.5. *Statistical taggers are not generalizable*

A text in which word tags are to be figured out must be similar enough to the reference corpus used to train the tagger, otherwise the quality of the results will be significantly impacted. As soon as a new word occurs in the text to be tagged, or a word occurs in the text to be tagged with a context different from those present in the reference corpus, the tagger will produce an error. To construct a tagger capable of producing correct results for any sentence in any type of text, it would be necessary to construct a reference corpus that contains all possible uses of each word in a language, which does not exist.

> Statistical POS taggers produce non-generalizable results.

The only reasonable alternative is to describe the whole vocabulary in a dictionary; this description would be solid, and if it contained errors or gaps these could be easily corrected, which would allow for the accumulation of data on the language.

1.5.2.6. *POS taggers use poor linguistic resources without admitting it*

It is wrong to claim that statistical taggers (as opposed to linguistic ones) do not use dictionaries or grammars: a tagger consults the reference corpus in search of potential tags for each word being tagged, and thus there is indeed an implicit dictionary in the reference corpus. This dictionary can easily be made explicit, by extracting the tagged words from the reference corpus and sorting them in alphabetical order. However, the resulting

12 See for example [DIC 05, GRE 10, VOL 11].

dictionary is of much lower quality than a "real" dictionary constructed by professional linguists [KUP 08].

We will see in Chapter 4 that dictionaries constructed by linguists distinguish, at the very minimum, the various uses of words; for example *reader* (+Hum) in *an avid reader* versus *reader* (-Hum) in *a magnetic reader; to give* (+ditransitive) in *Joe gave his pen to Lea* versus *to give* (+intransitive) in *The door finally gave*, etc. This basic level of precision bears no resemblance to the set of tags used by POS taggers.

> Taggers use very low-quality implicit dictionaries and produce very low-quality results.

1.5.2.7. *Rules of disambiguation are fundamentally incorrect*

The same is true for grammar: taggers do use a grammar, made up of the list of contexts of the words that occur in the reference corpus. For example, based on examples such as *I think that art is a worthwhile endeavor*, a tagger will typically decide that the word *that* should be tagged as "Conjunction" since it is preceded by the word *think* and followed by the word *art*, which is equivalent to the grammar rule below:

```
think that art → that/CONJUNCTION
```

The grammars used by taggers are automatically computed, based on sequences of successive words taken from the manually tagged reference corpus. These grammars are incorrect, and it is easy to produce counterexamples for each of their "rules". For example, in the sentence *I think that art is hung too low*, the word *that* is a determiner, not a conjunction, even though it appears between the words *think* and *art*.

In fact, the very principle of tagging – disambiguating each word, taking only its immediate context into account – is naïve in the extreme. The ten rules of disambiguation (called "patches") computed automatically by the tagger introduced in [BRI 92] are all incorrect: any linguist would consider it ridiculous to claim that in English, a capitalized word must be followed by a conjuguated verb in the preterite tense, or that a verbal form is necessarily in the past participle if the word *had* appears near it in a three-word context; for example in *I had to come*, *come* is not a past participle.

Taggers overlook the fundamental linguistic principle that sentences are structured objects, and that virtually any category of word can be inserted anywhere in a sentence without changing its structure. For example, after the verbal form *sees*, we might find an adjective (e.g. *Joe sees red apples*), an adverb (e.g. *Joe sees very well*), a determiner (e.g. *Joe sees that apple*), a coordinating conjunction (*Joe sees but says nothing*), a subordinating conjunction (*Joe sees that Lea is happy*), a noun (*Joe sees flowers*), a preposition (*Joe sees behind himself*), a pronoun (*Joe sees that*), a relative pronoun (*Joe sees where Lea is going*), or a verbal form (*Joe sees destroyed buildings*). In these conditions, disambiguating the word form *sees* (or any verbal form) on the basis of its contexts in a reference corpus – which necessarily possesses only a limited sample of all potential contexts – will produce a large number of errors.

Rules computed automatically by taggers are incorrect, and the grammars in which these rules are collected have neither the precision, nor the degree of generality, let alone any scientific value comparable to the grammars designed by linguists[13].

> Sentences are structured objects; any category of words can be inserted anywhere in a sentence without changing its structure. A grammar is not a collection of limited contexts of words and category.

1.5.2.8. *You cannot add apples to oranges*

This is a basic principle: you cannot add together (or calculate averages of) number of objects of different types. Yet this is what is done by most statistical taggers or automatic learning programs. For example, when a statistical program parses the two sentences below:

```
Joe/ProperName writes/Verb. He/Pronoun writes/Verb
a/Determiner novel/Noun.
```

it will deduce that the verb *to write* occurs sometimes without a complement (here: one time out of two), and sometimes with a direct object complement (here: one time out of two). Now if the same program parses the following text:

13 For an example of a real French grammar painstakingly created for the construction of an automatic software translation platform, see [SAL 99].

```
Joe/ProperName is/Aux running/Verb a/Determiner
test/Noun. Lea/ProperName is/Aux running/Verb
in/Preposition the/Determiner woods/Noun.
```

it will deduce, in the same way, that the verb *to run* sometimes occurs with a direct object complement, and sometimes with a prepositional complement… But this has nothing to do with it! In the first case, we can say that the verb *to write* has an optional direct object complement, but in the second case, it must be said that there are two homographic verbs *to run*[14]: the first is transitive, while the second is intransitive. We cannot simply add up the number of occurrences of the two verbs *to run* simply because they are spelled in the same way, just as we cannot say that the two forms *steel* and *steal* constitute the same linguistic unit just because they are pronounced in the same way.

> We cannot add up the number of occurrences of word forms that represent distinct elements of the vocabulary.

1.5.2.9. *Tagging does not allow for addressing syntactic or semantic ambiguities*

It is easy to construct examples of superficially identical word sequences that can be tagged correctly only after an overall syntactic analysis of the sentence, or even after a semantic analysis. A well known example, the form *like*, must be tagged "Verb" in the sentence *These horse flies like his arrow* and "Preposition" in the sentence *This horse flies like his arrow*. A simple examination of the immediate context of the word *like*, in other words, is not sufficient to determine whether it is a verb or a preposition. Taggers produce necessarily unreliable results in every case of ambiguity which only a syntactic or semantic analysis could solve.

> In the general case, word ambiguity cannot be resolved until a syntactic or semantic analysis has been carried out.

14 Statistical analysis programs are based on an absolute belief in the reliability of spelling. However, if one day a spelling reform were to decide that the noun *a bass* (in the sense of *a deep voice*) should be spelled *a basse* to distinguish it from the noun *a bass* (*the fish*), it would change nothing about the structure or meaning of texts, but it would change the "grammar rules" produced by statistical taggers.

1.5.2.10. *The scientific approach*

Even if the rules automatically produced by a tagger were correct, they would not have much scientific value; it is a bit like noting that in the work of Edgar Allan Poe, nouns containing an "a" and an "n" are all followed by a phrasal verb using particle "in": this remark has no generality, teaches us nothing about American literature, the English language, or Edgar Allan Poe himself, and is quite simply of no scientific interest.

Linguistics seek to understand how a language functions. Therefore even a "magical" statistical tool that could be used to build spectacular NLP applications but did not explain anything about the language is of little interest to us.

1.5.3. *Linguistic rather than stochastic analysis*

I am wary of the results produced by current stochastic methods in the field of NLP, especially when they are compared, on a *truly* level playing field, to those of linguistic methods.

I find it unfortunate that decision-makers in the NLP domain tend to favor stochastic methods that do not cause our understanding of language to advance a single step, to the detriment of projects aimed at building linguistic resources. Formalizing the lexical, morphological, syntactic, and distributional properties of the standard English vocabulary would require the work of a dedicated team, much smaller than the gigantic groups assigned to the construction of tagged corpora for statistical NLP applications (statistical tagging or translation). A project like this would be beneficial for the whole linguistic community and would enable the development of NLP software with unequalled precision. I hope this book will show that such a project is not just useful, but feasible as well.

1.6. Linguistic formalisms: NooJ

To formalize a language, we use mathematical models (referred to as *formalisms*). The vital question, posed by [CHO 57], is therefore: "Which mathematical model do we need in order to describe

languages?" Chomsky introduces a hierarchy of increasingly powerful grammars capable of describing increasingly complex linguistic phenomena[15], and hypothesizes that there is a "universal" formal model that can be used to describe any human language.

Paradoxically, this original issue has given rise to the creation of numerous incompatible linguistic formalisms. Today, linguists wishing to formalize linguistic phenomena can choose from among more than a dozen formalisms[16], including CCG, GPSG, HG, HFST, HPSG, LFG, LIG, OpenFst, RG, SFG, SFST, TAG, XFST, etc.

Each of these formalisms and their variants has individual strong and weak points. XFST (or HFST, or SFST) will be of more interest to morphologists, while GPSG, TAG or LFG are better suited to syntacticians, and semanticians will often use RG or SFG, while HPSG, as the most powerful formalism of the group, is typically used by theoreticians and linguists seeking to describe phenomena at the limit of linguistics, such as anaphora. Unfortunately, none of these tools makes it possible to describe in a simple manner the wide variety of linguistic phenomena (often quite trivial) that come into play when "real" texts are to be analyzed (such as journalistic texts or novels), and it is not possible to combine them since their formalisms, their development environments, and IT tools are largely incompatible[17].

The search for a single formalism capable of addressing all types of linguistic phenomena in all languages does not fall within the parameters of the project described in this book; our goal is rather to describe the largest number of linguistic phenomena in their diversity, using the best-suited tools in order to construct simple descriptions, i.e. those that are the simplest to develop, understand, accumulate, share, and manage.

15 We will look in detail at the Chomsky-Schützenberger hierarchy in Chapter 5.
16 See the Internet links at the end of this chapter.
17 An exception that proves the rule is the Urdu parser in the ParGram project (based on LFG) which used XFST as a preprocessing tool to transliterate Urdo or Devanagari characters and recognize repetitions; see [BÖG 07]. However, the XFST descriptions did not interact with those of LFG.

Figure 1.7. *A single tool for formalization: NooJ*

It was by abandoning the idea of a universal formalism that I designed the linguistic platform NooJ[18]. With NooJ, orthographic variation in Chinese is not described with the same tools as morphological derivation in Arabic. Neither is agglutination described with the same tools as inflection, which is not described in the same way as syntax, etc. NooJ guarantees high integration of all levels of description thanks to compatible notations and a unified representation for all linguistic analysis results (the text annotated structure or TAS, see Chapter 10), enabling different analyzers at different linguistic levels to communicate with one another. For example, the following transformation:

Lea donates her blood → *Lea is a blood donor*

brings various levels of linguistic phenomena into play: it is necessary to verify that the verb *to donate* will admit a nominalization (thanks to a lexical property); we must describe the inflectional rule that produces *to donate* from *donates*, the derivational rule that produces *donor* from *donate*, a distributional constraint to verify that *Lea* is a human subject and that *blood* falls under the category of parts of the body (a person donates his/her heart, a kidney, etc.), a syntactic rule to verify that *Lea* is the subject of *donates*, and then a restructuration rule to move the noun *blood* from the role of direct object complement to the role of complement of the noun *donor*. No single formalism could make it possible to describe all these linguistic phenomena in a simple way.

18 See [SIL 03a]. NooJ is a linguistic development environment that operates on Windows, Mac OSX, LINUX and Unix, available at no cost at the website www.noo4nlp.net. An open source environment, it is supported by the European project METANET-CESAR.

Thanks to the suitability (and thus simplicity) of each individual tool used to describe each level of linguistic phenomenon, it is now possible to construct, test, accumulate and combine linguistic resources of diverse types, and this for many languages. Through the integration of all levels of analysis in the TAS (see Chapter 10), we will see that it becomes possible to carry out complex analyses of a text by performing a series of different analyses, each of them quite simple[19].

In this book, we will use the NooJ notation to describe linguistic phenomena in their diversity. Using only one notation will simplify their understanding, as the reader does not need to master half a dozen different formalisms to understand how to formalize all types of linguistic phenomena. NooJ also has the advantage of being very easy to use; one can learn NooJ in just a few hours, and even become an "expert" after a week of training[20].

It goes without saying that each of the linguistic phenomena mentioned in this book can be described with one of the formalisms traditionally used in linguistics; for example, XFST could be used to construct the morphological parser introduced in Chapter 11, GPSG to construct syntax trees, LFG to describe agreement constraints such as those shown in Chapter 12, and HPSG to formalize the transformations discussed in Chapter 13.

1.7. Conclusion and structure of this book

The goal of the project depicted in this chapter is to describe natural languages very precisely and mathematically, or more specifically, to formalize the set of sentences that may appear in written texts.

19 NooJ shares several characteristics with other integrated toolboxes such as the General Architecture for Text Engineering (GATE) and the Stanford Core NLP. It consists of independent modules applied in cascade (or pipeline) in a bottom-up approach that communicate via a text annotation structure (the TAS, see Chapter 10). The main differences between NooJ and these NLP toolboxes are that NooJ is a pure linguistic tool (for instance, there is no statistical tagger in NooJ) and that all its modules are formalized via descriptive grammars rather than implemented via software programs. In other words, Nooj follows a purely descriptive, rather than algorithmic approach.

20 A growing number of NooJ users are not linguists, but rather historians, scholars of literature, psychologists, and sociologists, who typically use NooJ to extract "interesting" information from their corpora.

Although the number of sentences in a language is infinite, it is nevertheless possible to describe it, starting by using a finite number of basic linguistic elements (letters, morphemes, or words), which I will do in Part 1 of this book. Chapter 2 shows how to formalize the alphabet of a language; Chapter 3 discusses how to delineate its vocabulary; and Chapter 4 shows how to formalize a vocabulary using electronic dictionaries.

Next we must equip ourselves with mechanisms for describing how those basic linguistic elements combine to construct higher-level elements (word forms, phrases or sentences). Part 2 of this book introduces the concepts of *formal language*, *generative grammar*, and *machines*. I will introduce these concepts as well as the Chomsky-Schützenberger hierarchy in Chapter 5, while Chapters 6 to 9 present the four types of languages/grammars/machines.

Part 3 of this book (Chapters 10 to 13) is dedicated to the automatic linguistic analysis of texts. In Chapter 10, I will introduce the TAS used to represent, in a unified manner, the results produced by all linguistic analyses. Chapter 11 is devoted to automatic lexical analysis. Chapter 12 introduces two types of syntactic analysis: local analysis and structural analysis. Chapter 13 presents an automatic transformational analyzer, which can be seen as a linguistic semantic analyzer (that is, providing an analysis of meaning based solely on language, without real-world knowledge or inference).

1.8. Exercises

1) Based on the model of the definition of $\mathbb{N}$ seen in section 1.1, characterize the set $\mathbb{D}$ that contains all the decimal numbers.

2) Take the sentence: *His great uncle was let go on the spot*. Describe the linguistic analyses of this sentence in informal lexical, morphological, syntactic, and semantic terms.

3) Consider the text: *Lea invited Ida for dinner. The graduate student brought a bottle of wine*. How can we figure out to whom the noun *graduate student* is referring? Can this calculation be made using linguistic analyses?

4) How can we improve the translation of the first sentence in Figure 1.5, using only a bilingual dictionary, and without any grammar?

5) Construct the dictionary implicitly given in the reference corpus of Figure 1.6. Compare the content of this dictionary with the content of an editorial dictionary.

1.9. Internet links

The Wikipedia page for the field of linguistics: en.wikipedia.org/wiki/Linguistics

The Wikipedia page for Natural Language Processing: en.wikipedia.org/wiki/Natural_language_processing

An NLP course taught by Dan Jurafsky and Chris Manning, Stanford University, is available on YouTube atwww.youtube.com/watch?v=nfoudtpBV68.

The *Jeopardy!* program of April 11, 2001, which the computer IBM Watson won in spectacular fashion, has been the subject of numerous documentaries, for example: www.youtube.com/watch?v=5Gpaf6NaUEw.

To test a few pieces of machine translation software, such astranslate.google.com, www.systranet.com/translate, www.reverso.net, www.freetranslation.com, etc. you can enter a short paragraph in English, translate it into another language, copy and paste the result, and retranslate that into English.

There are many automatic text-tagging software platforms; see en.wikipedia.org/wiki/Part-of-speech_tagging. Tagging software requires a reference corpus, such as Penn Treebank which can be found at www.cis.upenn.edu/~treebank. Many annotated text corpora can be obtained from the Linguistic Data Consortium at www.ldc.upenn.edu.

Two integrated toolboxes used for building NLP software applications (mostly taggers and annotators) are the General Architecture for Text Engineering (Gate) (gate.ac.uk) and the Stanford CoreNLP: nlp.stanford.edu/software/corenlp.shtml.

The formalisms traditionally used by linguists are:

– CCG (Combinatory Categorial Grammar): groups.inf.ed.ac.uk/ccg/

– HFST (Helsinki Finite-State Transducer): www.ling.helsinki.fi/kielite knologia/ tutkimus/hfst

– HG (Head Grammar): en.wikipedia.org/wiki/Head_grammar

– HPSG (Head-Driven Phrase Structure Grammar): hpsg.stanford.edu

– LFG (Lexical Functional Grammar): www.essex.ac.uk/linguistics/ external/LFG

– LIG (Linear Indexed Grammar): www.inf.ed.ac.uk/teaching/courses/ inf2a/slides/2011_inf2a_L21_slides.pdf

– OpenFST: www.openfst.org

– RG (Relational Grammar): en.wikipedia.org/wiki/Relational_grammar

– SFST (Stuttgart Finite-State Transducer): code.google.com/p/ cistern/wiki/SFST

– TAG (Tree-adjoining grammar): www.cis.upenn.edu/~xtag/tech-report/node6.html

– XFST (Xerox Finite-State Tool): www.cis.upenn.edu/~cis639/docs/ xfst.html

The linguistic development environment that will be used to describe all the linguistic phenomena in this book is NooJ, a free and open-source software platform made available under GPL license by the European Community program META-SHARE; see www.nooj4nlp.net.