

Chapter 1

Introduction

"Linguistics accordingly works continuously with concepts forged by grammarians without knowing whether or not the concepts actually correspond to the constituents of the system of language. But how can we find out? And if they are phantoms, what realities can we place in opposition to them?" — de Saussure [211, p. 110]

Abstract The Structure Discovery paradigm for Natural Language Processing is introduced. This is a framework for learning structural regularities from large samples of text data, and for making these regularities explicit by introducing them in the data via self-annotation. In contrast to the predominant paradigms, Structure Discovery involves neither language-specific knowledge nor supervision and is therefore independent of language, domain and data representation. Working in this paradigm means to set up discovery procedures that operate on raw language material and iteratively enrich the data by using the annotations of previously applied Structure Discovery processes. Structure Discovery is motivated and justified by discussing this paradigm along Chomsky's levels of adequacy for linguistic theories. Further, the vision of the complete Structure Discovery Machine is sketched: A series of processes that allow analysing language data by proceeding from generic to specific. At this, abstractions of previous processes are used to discover and annotate even higher abstractions. Aiming solely to identify structure, the effectiveness of these processes is judged by their utility for other processes that access their annotations and by measuring their contribution in application-based settings. A data-driven approach is also advocated when defining these applications, proposing crowdsourcing and user logs as a means to widen the data acquisition bottleneck.

1.1 Structure Discovery for Language Processing

In the past, Natural Language Processing (NLP) has always been based on *explicit* or *implicit* use of linguistic knowledge. Explicit rule based approaches prevail in classical linguistic applications, while machine learning algorithms use implicit knowledge for generating linguistic annotations.

The question behind this work is: how far can we go in NLP without assuming any linguistic knowledge? How much effort in annotation and resource building is needed for what level of sophistication in natural language analysis?

Working in what I call the *Structure Discovery* (SD) paradigm, the claim being made here is that the required knowledge can largely be acquired by knowledge-free and unsupervised methods. By employing knowledge about language universals (cf. Chapter 3) it is possible to construct Structure Discovery processes, which operate on a raw text corpus¹ in an iterative fashion to unveil structural regularities in the text and to make them explicit in a way that further SD processes can build upon it.

A frequent criticism on work dealing with unsupervised methods in NLP is the question: "Why not take linguistic knowledge into account?" The simple answer to this is that for many languages and applications, the appropriate linguistic knowledge just is not available. While annotated corpora, classification examples, sets of rules and lexical semantic word nets of high coverage exist for English, this does not reflect the situation for most of major world languages. Further, handmade and generic resources often do not fit the application domain, whereas resources created *from and for* the target data inherently do not suffer from these discrepancies.

Structure in this context is any kind of automatically acquired annotation that relates elements of language along arbitrary kinds of similarity. It is not restricted to a single language level, but encompasses labels with the scope of e.g. sentences, clauses, words or substrings. Since the processes that discover and mark structural regularities are not allowed to choose from a predefined inventory of labels, the names of the labels are meaningless and receive their interpretation merely through the elements marked by them.

Unlike other works that proceed by teaching the machine directly how to solve certain tasks — be it by providing explicit rules or implicitly by training the machine on handcrafted examples — the scope of this work is the unsupervised, knowledge-free acquisition of structural regularities in language. Unsupervised means that no labelled training material is provided as input. The machine is exposed to language only, without being told what its output should look like. Knowledge-free means that no knowledge about the specific language, such as e.g. word order constraints or a list of personal pronouns, is given to the system. The work of a Structural Discovery engineer is rather to provide a suitable collection of natural language data and to set up procedures that make use of it.

All knowledge about how to conduct this augmentation of structure is encoded procedurally in methods and algorithms. This keeps the paradigm entirely language independent and applicable without further effort to all human languages and sub-languages for which data is available. Given the fact that several thousand human languages are spoken in the world and given the ongoing specialisation in science, production and culture, which is reflected in respective sub-languages or domains, this paradigm provides a cheaper, if not the only way of efficiently dealing with this variety.

¹ the terms text, text data, corpus, text corpus, language and language data are used interchangeably throughout this work.

Shifting the workload from creating rich resources manually to developing generic, automatic methods, a one-size-fits-all solution needing only minimal adaptation to new domains and other languages comes into reach. While the final task is defined by the application, and data has still to be collected that defines the target behaviour of the system, the core point of Structure Discovery is to fully automatise the natural language preprocessing steps that are necessary to build the application.

In the remainder of this section, the paradigms followed by the fields of Computational Linguistics (CL) and Natural Language Processing (NLP) are shortly contrasted after laying out the SD paradigm in more detail. Further, the benefits and drawbacks of knowledge-free as opposed to knowledge-intensive approaches, as well as degrees of supervision are elaborated on and the SD paradigm is compared to other paradigms.

1.1.1 Structure Discovery Paradigm

The Structure Discovery (SD) paradigm is the research line of algorithmic descriptions that find and employ structural regularities in natural language data. The goal of SD is to enable machines to grasp regularities, which are manifold and necessary in data used for communication, politics, entertainment, science and philosophy, purely from applying operationalised procedures on data samples. Structural regularities, once discovered and marked as such in the data, allow an abstraction process by subsuming structural similarities of basic elements. These complex bricks constitute the building block material for meta-regularities, which may themselves be subject of further exploration.

The iterative methodology of the SD paradigm is laid out in [Figure 1.1](#). It must be stressed that working in the SD paradigm means to proceed in two directions: using the raw data for arriving at generalisations and using these generalisations to structurally enrich the data. While the first direction is conducted by all clustering approaches, the second direction of feeding the results back into the data to perform self-annotation is only rarely encountered.

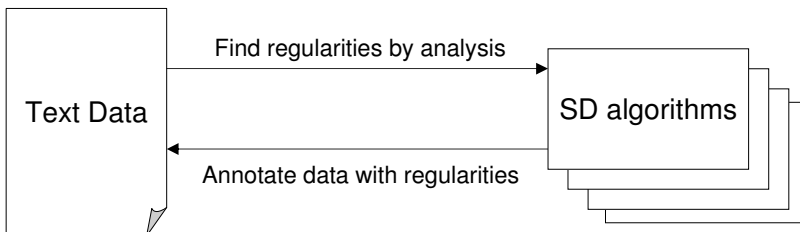


Fig. 1.1 Iterative methodology of the Structure Discovery paradigm: SD algorithms find and annotate regularities in text data, which can serve as input for further SD algorithms

Unlike other schools that provide knowledge of some sort to realise language processing, a system following the SD paradigm must arrive at an adequate enrichment of language data with the instances of the discovered structures, realising self-annotation of the data, as merely the data in its raw form determines the kind of structures found and instances identified thereof.

Solely working on algorithmically discoverable structures means to be consequently agnostic to linguistic theories. It is not possible for machines to create proposals for analysis based on intuition and introspection. Rather, the grammarian's knowledge can stimulate discovery process formulation. The native speaker's intuition about particular languages is replaced by a theory-independent intuition of how to discover structure. While as well created intellectually, the utility of the new discovery process can immediately be judged and measured. Further, it is applicable for all data exhibiting similar structure and thus more general.

1.1.2 Approaches to Automatic Language Processing

The discipline of automatically processing natural language is split into two well-established subfields that are aimed at slightly different goals. Computational Linguistics (CL) is mainly influenced by linguistic thinking and aims at implementing linguistic theories following a *rationalist* approach. Focus is set on linguistic problems rather than on robust and efficient processing. Natural Language Processing (statistical NLP, in the definition of Manning and Schütze [158]), on the other hand, is not linked to any linguistic theory. The goal of NLP is not understanding of language structure as an end in itself. Rather, knowledge of language structure is used to build methods that help in processing language data. The criterion of NLP is the performance of the system, not the adequacy of the representation to human language processing, taking a pragmatic but theoretically poorly founded view. Regarding current research, the two fields are not easily separated, as they influence and fertilise each other, so there is rather a continuum than a sharp cutting edge between the two.

The history of automatic treatment of natural languages was dominated consecutively by two major paradigms: rule-based and statistical approaches. Starting with the realm of computers, rule-based approaches tackled more and more problems of language processing. The leitmotif was: given enough time to develop more and more sophisticated rules, eventually all phenomena of language will be encoded. In order to operationalise the application of grammar formalisms, large systems with several thousand grammar rules were built. Since these rules interact with each other, the process of adding sensible rules gets slower the more rules are already present, which makes the construction of rule-based systems expensive. What is inherent of the top-down, introspection-driven construction of rule-based systems is that they can only work on the subset of language covered by the rules. History has shown that the size of this subset remained far from getting close to full coverage, resulting in only a fraction of sentences being processable.

Since the advent of large machine-readable text corpora starting with the Brown Corpus [103] and the computing power necessary to handle them, statistical approaches to language processing received increased interest. By basing decisions on probabilistic models instead of rules, this *empiricist* approach early showed to be capable of reaching higher accuracy on language processing tasks than the rule-based approach. The manual labour in statistical methods is shifted from instructing the machines directly by rules how to process the data to labelling training examples that provide information on how a system's output should look like. At this point, more training means a richer basis for the induction of probabilistic models, which in turn leads to better performance.

For understanding language from a linguistic point of view and testing grammar theories, there does not seem to be another way than the rule-based approach. For building applications, however, statistical methods proved to be more robust and to scale better, which is probably best contrasted for Machine Translation by opposing Martin Kay's essay "Machine Translation: The Disappointing Past and Present" [137] to Franz Josef Och's talk "Statistical Machine Translation: The Fabulous Present and Future" [187].

Completely opposed to the rule-based approach, the work described in this volume takes the statistical approach a step further by not even allowing implicit knowledge to be provided for training.

1.1.3 Knowledge-intensive and Knowledge-free

Another dimension, along which it is possible to classify language processing methods, is the distinction between knowledge-intensive and knowledge-free approaches, see also [42]. Knowledge-intensive approaches make excessive use of language resources such as dictionaries, phrase lists, terminological resources, name gazetteers, lexical-semantic networks such as WordNet [178], thesauri such as Roget's Thesaurus [204], ontologies and the like. As these resources are necessarily incomplete (all resources leak), their benefit will only extend to a certain point. Additional coverage can only be reached by substantial enlargement of the resource, which is often too much of a manual effort.

But knowledge-intensiveness is not only restricted to explicit resources: the rules in a rule-based system constitute a considerable amount of knowledge, just like positive and negative examples in machine learning.

Knowledge-free methods seek to eliminate human effort and intervention. The human effort is not in specifying rules or examples, but in the method itself, lending the know-how by providing discovery procedures rather than presenting the knowledge itself. This makes knowledge-free methods more adaptive to other languages or domains, overcoming the brittleness of knowledge-intensive systems when exposed to an input substantially different from what they were originally designed for.

Like above, there rather is a continuum than a sharp border between the two ends of the scale. Methods that incorporate only little human intervention are sometimes labelled knowledge-weak, combining the benefits of not having to prepare too much knowledge with obtaining good results by using available resources.

1.1.4 Degrees of Supervision

Another classification of methods, which is heavily related to the amount of knowledge, is the distinction between supervised, semi-supervised, weakly supervised and unsupervised methods.

- In *supervised* systems, the data as presented to a machine learning algorithm is fully labelled. That means: all examples are presented with a classification that the machine is meant to reproduce. For this, a classifier is learned from the data, the process of assigning labels to yet unseen instances is called classification.
- In *semi-supervised* systems, the machine is allowed to additionally take unlabelled data into account. Due to a larger data basis, semi-supervised systems often outperform their supervised counterparts using the same labelled examples (see [254] for a survey on semi-supervised methods and [210] for a summary regarding NLP and semi-supervision). The reason for this improvement is that more unlabelled data enables the system to model the inherent structure of the data more accurately.
- Bootstrapping, also called self-training, is a form of learning that is designed to use even less training examples, therefore sometimes called *weakly-supervised*. Bootstrapping (see [23] for an introduction) starts with a few training examples, trains a classifier, and uses thought-to-be positive examples as yielded by this classifier for retraining. As the set of training examples grows, the classifier improves, provided that not too many negative examples are misclassified as positive, which could lead to deterioration of performance.
- *Unsupervised* systems are not provided any training examples at all and conduct clustering. This is the division of data instances into several groups, (see [158, Ch. 14] and [20] for an overview of clustering methods in NLP). The results of clustering algorithms are data driven, hence more 'natural' and better suited to the underlying structure of the data. This advantage is also its major drawback: without a possibility to tell the machine what to do (like in classification), it is difficult to judge the quality of clustering results in a conclusive way. But the absence of training example preparation makes the unsupervised paradigm very appealing.

To elaborate on the differences between knowledge-free and unsupervised methods, consider the example of what is called unsupervised (also called knowledge-based) word sense disambiguation [cf. 251]. Word sense disambiguation (WSD) is the process of assigning one of many possible senses to ambiguous words in the text. This can be done supervisedly by learning from manually tagged examples. In the termi-

nology of Senseval-3 [172], an unsupervised WSD system decides the word senses merely based on overlap scores of the word’s context and a resource containing word sense definitions, e.g. a dictionary or WordNet. Such a system is unsupervised, as it does not require training examples, but knowledge-intensive due to the provision of the lexicographic resource.

1.1.5 *Contrasting Structure Discovery with Previous Approaches*

To contrast the paradigm followed by this work with the two predominant paradigms of using explicit or implicit knowledge, Table 1.1 summarises their main characteristics.

Paradigm	classical CL	statistical NLP	SD
Approach	rule-based	statistics	statistics
Direction	top-down	bottom-up	bottom-up
Knowledge Source	manual resources	manual annotation	–
Knowledge Intensity	knowledge-intensive	knowledge-intensive	knowledge-free
Degree of Supervision	unsupervised	supervised	unsupervised
Corpus required	–	annotated text	raw text

Table 1.1 Characteristics of three paradigms for the computational treatment of natural language

Various combinations of these paradigms lead to hybrid systems, e.g. it is possible to construct the rules needed for CL by statistical methods, or to build a supervised standard NLP system on top of an unsupervised, knowledge-free system, as conducted in Section 6.9. As already discussed earlier, semi-supervised learning is located in between statistical NLP and the SD paradigm.

1.2 Relation to General Linguistics

Since the subject of examination in this work is natural language, it is inevitable to relate the ideas presented here to linguistics. Although in this book, neither one of the dominating linguistic theories is implemented nor something that would be dubbed ‘linguistic theory’ by linguists is proposed, it is still worthwhile looking at those ideas from linguistics that inspired the methodology of unsupervised natural language processing, namely linguistic structuralism and distributionalism. Further, the framework shall be examined along desiderata for language processing systems, which were formulated by Noam Chomsky.

Serving merely to outline the connection to linguistic science, this section does by no means raise a claim for completeness on this issue. For a more elaborate discussion of linguistic history that paved the way to the SD paradigm, see [42].

1.2.1 Linguistic Structuralism and Distributionalism

Now, the core ideas of linguistic structuralism will be sketched and related to the SD paradigm. Being the father of modern linguistics, de Saussure [211] introduced his negative definition of meaning: the signs of language (think of linguistic elements such as words for the remainder of this discussion) are solely determined by their relations to other signs, and not given by a (positive) enumeration of characterisations, thereby harshly criticising traditional grammarians. This is to say, language signs arrange themselves in a space of meanings and their value is only differentially determined by the value of other signs, which are themselves characterised differentially.

De Saussure distinguishes two kinds of relations between signs: syntagmatic relations that hold between signs in a series in present (e.g. neighbouring words in a sentence), and associative relations for words in a "potential mnemonic series" [211, p. 123]. While syntagmatic relationships can be observed from what de Saussure calls *langage* (which corresponds to a text corpus in terms of this work), all other relationships subsumed under 'associations' are not directly observable and can be individually different. A further important contribution to linguistic structuralism is attributed to Zellig Harris [124, 123], who attempts to discover some of these associative or paradigmatic relations. His *distributional hypothesis* states that words of similar meanings can be observed in similar contexts, or as popularised by J. R. Firth: "You shall know a word by the company it keeps!" [99, p.179]². This quote can be understood as the main theme of *distributionalism*: determining the similarity of words by comparing their contexts.

Distributionalism does not look at single occurrences, but rather at a word's distribution, i.e. the entirety of contexts (also: global context) it can occur in. The notion of context is merely defined as language elements related to the word; its size or structure is arbitrary and different notions of context yield different kinds of similarities amongst the words sharing them. The consequences of the study of Miller and Charles [179] allow to operationalise this hypothesis and to define the similarity of two signs as a function over their global contexts: the more contexts two words have in common, the more often they can be exchanged, and the more similar they are. This immediately gives rise to discovery procedures that compare linguistic elements according to their distributions, as conducted in later chapters of this book.

² Ironically, Firth did not mean to pave the way to a procedure for statistically finding similarities and differences between words. He greatly objected to de Saussure's views and clearly preferred the study of a restricted language system for building a theory, rather than using discovery procedures for real, unrestricted data. In his work, the quote refers to assigning correct meanings to words in habitual collocations.

Grouping sets of mutually similar words into clusters realises an abstraction process, which allows generalisation for both words and contexts via class-based models [cf. 49]. It is this mechanism of abstraction and generalisation, based on contextual clues, that allows the adequate treatment and understanding of previously unseen words, provided their occurrence in well-known contexts.

1.2.2 Adequacy of the Structure Discovery Paradigm

This section aims at providing theoretical justification for bottom-up discovery procedures as employed in the SD paradigm. This is done by discussing them along the *levels of adequacy* for linguistic theories, set up in [59], and examining to what extent this classification applies to the procedures discussed in this work. For this, Chomsky's notions of linguistic theory and grammar have to be briefly sketched. For Chomsky, a (generative) grammar is a formal device that can generate the infinite set of grammatical (but no ungrammatical) sentences of a language. It can be used for deciding the grammaticality of a given sentence [see also 58]. A linguistic theory is the theoretical framework, in which a grammar is specified. Chomsky explicitly states that linguistic theory is only concerned with grammar rules that are identified by introspection, and rejects the use of discovery procedures of linguistic regularities from text corpora, since these are always finite and cannot, in his view, serve as a substitute for the native speaker's intuitions.

Having said this, Chomsky provides a hierarchy of three levels of adequacy to evaluate grammars.

- A grammar with *observational adequacy* accounts for observations by exhaustive enumeration. Such "item-and-arrangement grammars" [59, p. 29] can decide whether a given sentence belongs to the language described by the grammar or not, but does not provide any insights into linguistic theory and the nature of language as such.
- A higher level is reached with *descriptive adequacy*, which is fulfilled by grammars that explain the observations by rules that employ "significant generalizations that express underlying regularities of language" [59, p. 63].
- *Explanatory Adequacy* is the highest level a grammar can reach in this classification. Grammars on this level provide mechanisms to choose the most adequate of competing descriptions, which are equally adequate on the descriptive level. For this, "it aims to provide a principled basis, independent of any particular language" [59, p. 63].

According to Chomsky, the levels of descriptive and explanatory adequacy can only be reached by linguistic theories in his sense, as only theoretic means found by introspection based on the native speaker's intuition can perform the necessary abstractions and meta-abstractions. Criticising exactly this statement is the subject of the remainder of this section.

When restricting oneself to a rule-based description of universal language structure like Chomsky does, there does not seem to be any other option than proceeding in an introspective, highly subjective and principally incomplete way: as already Sapir [209, p. 38] stated: "all grammars leak", admitting the general defect of grammar theories to explain the entirety of language phenomena. Especially when proceeding in a top-down manner, the choice "whether the concept [an operational criterion] delimits is at all close to the one in which we are interested" [59, p. 57] is subjective and never guaranteed to mirror linguistic realities. But when dropping the constraint on rules and attributing explanatory power to bottom-up discovery procedures, the levels of adequacy are also applicable to the SD framework.

Admitting that operational tests are useful for soothing the scientific conscience about linguistic phenomena, Chomsky errs when he states that discovery procedures cannot cope with higher levels of adequacy [59, p. 59]. By using clustering procedures as e.g. described in [49] and in Chapter 4, abstraction and generalisation processes are realised that employ the underlying structure of language and thus serve as algorithmic descriptions of language phenomena. Further, these class-based abstractions allow the correct treatment of previously unobserved sentences, and a system as described in Section 6 would clearly attribute a higher probability (and therefore acceptability) to the sentence "colorless green ideas sleep furiously" than to "furiously sleep ideas green colorless" based on transition probabilities of word classes (examples taken from [59, p. 57]), see also [186] on this issue.

Unlike linguistic theories, the systems equipped with these discovery procedures can be evaluated either directly by inspection or indirectly by measuring their contribution to an application. It is therefore possible to decide for the most adequate, best performing discovery procedure amongst several available ones. Conducting this for various languages, it is even possible to identify to what extent discovery procedures are language independent, and thus to arrive at explanatory power, which is predictive in that sense that explanatory adequate procedures can be successfully used on previously unseen languages.

The great advantage of discovery procedures is that, once defined algorithmically, they produce abstractions that are purely based on the data provided, being more objective and conclusive than rules found by introspection. While simply not fitting in the framework of linguistic theories, they are not phantoms but realities of language, so a complete description of language theory should account for them [see also 1].

In terms of quantity and quality, the goals of linguistic theory and unsupervised discovery procedures are contrary to one another. Linguistic theory aims at accounting for most types of phenomena irrespective of how often these are observed, while application-based optimisation targets at an adequate treatment of the most frequent phenomena. Therefore, in applying unsupervised discovery procedures, one must proceed quantitatively, and not qualitatively, which is conducted in this work at all times.

1.3 Similarity and Homogeneity in Language Data

1.3.1 Levels in Natural Language Processing

Since the very beginning of language studies, it is common ground that language manifests itself by interplay of various levels. The classical level hierarchy in linguistics, where levels are often studied separately, is the distinction of [see e.g. 116] phonological, morphological, syntactic, semantic and pragmatic level. Since this work is only concerned with digitally available written language, levels that have to do with specific aspects of spoken language like phonology are not considered here. Merely considering the technical data format for the text resources used here, basic units and levels of processing in the SD paradigm are [cf. 126]:

- character level
- token level
- sentence level
- document level

The character level is concerned with the alphabet of the language. In case of digitally available text data, the alphabet is defined by the encoding of the data and consists of all valid characters, such as letters, digits and punctuation characters. The (white)space character is a special case, as it is used as delimiter of tokens. Characters as the units of the character level form words by concatenation. The units of the token level are tokens, which roughly correspond to words. Hence, it is not at all trivial what constitutes a token and what does not. The sentence level considers sentences as units, which are concatenations of tokens. Sentences are delimited by sentence separators, which are given by full stop, question mark and exclamation mark. Tokenisation and sentence separation are assumed to be given by a pre-processing step outside of the scope of this work.³

A clear definition is available for documents, which are complete texts of whatever content and length, i.e. web pages, books, newspaper articles etc.

When starting to implement data-driven acquisition methods for language data, only these units can be used as input elements for SD processes, since these are the only directly accessible particles that are available in raw text data.

It is possible to introduce intermediate levels: morphemes as subword units, phrases as subsentential units or paragraphs as subdocument units. However, these and other structures have to be found by the SD methodology first, so they can be traced back to the observable units.

Other directly observable entities, such as hyperlinks or formatting levels in web documents, are genre-specific levels that might also be employed by SD algorithms, but are not considered for now.

³ for experiments in later chapters, the tools of the Leipzig Corpora Collection (LCC, [198]) were used throughout for tokenisation and sentence separation.

1.3.2 *Similarity of Language Units*

In order to group language units into meaningful sets, they must be compared and similarity scores for pairs of units need to be assigned. Similarity is determined by two kinds of features: *surface features* and *contextual features*. Surface features are obtained by only looking at the unit itself, i.e. tokens are characterised on their surface by the letters and letter combinations (such as character N -grams) they contain, sentences and documents are described with the tokens they consist of. Contextual features are derived by taking the context of the unit into consideration.

The context definition is arbitrary and can consist of units of the same and units of different levels. For example, tokens or character N -grams are contextually characterised by other tokens or character N -grams preceding them, sentences are maybe similar if they occur in the same document, etc. Similarity scores based on both surface and contextual features require exact definition and a method to determine these features, as well as a formula to compute the similarity scores for pairs of language units.

Having computational means at hand to compute similarity scores, language units can eventually be grouped into homogeneous sets.

1.3.3 *Homogeneity of Sets of Language Units*

While the notion of language unit similarity provides a similarity ranking of related units with respect to a specific unit, a further abstraction mechanism is needed to arrive at classes of units that can be employed for generalisation. Thus, it is clear that a methodology is needed for grouping units into meaningful sets. This is realised by clustering, which will be discussed in depth in Chapter 4. In contrast to the similarity ranking centred on a single unit, a cluster consists of an arbitrary number of units. The advantage is that all cluster members can be subsequently subsumed under the cluster, forming a new entity that can give rise to even more complex entities.

Clustering can be conducted based on the pairwise similarity of units, based on arbitrary features. In order to make such clustering and at the same time generalisation processes successful, the resulting sets of units must exhibit *homogeneity* in some dimensions. Homogeneity here means that the cluster members agree in a certain property, which constitutes the abstract concept of the cluster. Since similarity can be defined along many features, it is to be expected that different dimensions of homogeneity will be found in the data, each one covering only certain aspects, e.g. on syntactic, semantic or morphological levels. Homogeneity is, in principle, a measurable quantity and expresses the plausibility of the grouping. In reality, however, this is very often difficult to quantify, thus different groupings will be judged on their utility in the SD paradigm: on the one hand by their ability to generalise in a way that further structure can be discovered with these abstract concepts forming the building blocks, on the other hand by their utility as features in task-driven applications.

1.4 Vision: The Structure Discovery Machine

The remainder of this chapter is dedicated to an outline of the ultimate goal of SD: a set of algorithmic procedures that encompasses the discovery of the entirety of structure that can be discovered in a data-driven way. Viewing the practical results in Chapters 5, 6 and 7 as being only a starting point, I will now envision a number of capabilities of such a system and discuss them along the usual pipeline of processing steps for language processing. The 'traditional' terms shall serve as an aid for imagination — without doubt, the discovery procedures will not reproduce theoretically pure sub-levels, as indicated above. Nevertheless, linguistic theory can play the role of a source of what is necessary and which phenomena are to be accounted for.

When exhibited to language data, the Structure Discovery Machine (SDM) identifies the basic word and sentence units it will operate on and finds a suitable representation — no matter whether the SDM is exposed to text, speech or other encodings of language. Already here, an abstraction process is involved that groups e.g. different phonetic variants in speech or different character sets in written language. Then, different languages are identified in the stream and the corresponding parts are grouped (see Section 5). In a similar way, domain-specific subsets of the monolingual material are marked as such, e.g. by techniques as employed in document clustering (see [231] for an overview).

For each language, a syntactic model is set up that encompasses parts of speech (cf. Section 6) for basic word units, chunking to phrasal units of several words [as in e.g. 67] and syntactic dependencies between basic and complex units⁴ such as in [140; 36; 188, inter al.]. This requires a procedure handling derivation, inflection and compounding of units, realised by a component similar to those compared in the MorphoChallenge [146], or more recently [113]. Contextual analysis of content-bearing units allows to hypothesise different meanings for units with semantically ambiguous use and to disambiguate their occurrences (cf. Section 7). Having circumnavigated the pitfalls of lexical ambiguity, units are classified according to their semantic function and relation. Semantic classes have been previously learned in [e.g. 53; 21; 235]; for semantic relations, [238] shows ways how to extract them from massive corpora. The works of Davidov and Rappoport [77] and Davidov et al. [78] illustrate methods on how to extract sets of semantically similar words and their typical relations from web data. Reoccurring relations between units of similar or different semantic function will be marked as such, serving as something similar to FrameNet [14] annotations. Coreference has been successfully learnt in an unsupervised way by [120], information extraction with self-learned templates is laid out in [56].

In its highest and purest form, the SDM will not even be engineered by humans plugging together discovery procedures based on their intuitions, but will self-assemble by trying out interplays of a parameterisable inventory of procedures, thus optimising the overall structural description in terms of generalisation power. This,

⁴ cf. approaches to parsing, such as HPSG [193], LFG [128] and dependency parsing [184].

however, is a much larger project than outlined in this book, and raises several yet unanswered research questions.

Output of the SDM is a multidimensional annotation of the raw input data with labels that denote abstractions of structurally similar phenomena. Some kinds of annotations are orthogonal to each other, others can be arranged hierarchically and many will be dependent on other annotations. The scope of annotations is not limited, ranging from the most basic to the most complex units. This holistic approach to data-driven self-annotation rather overgenerates and is not restricted to a small set of structural phenomena. To determine which generalisations are useful, these have to be tested for utility in task-based evaluations, see next section.

Applications of these annotations are twofold: firstly, they can be employed as features in machine learning for a task-based adaptation: stemming, parts-of-speech tagging, chunking, named entity recognition, information extraction and parsing in their present form will greatly benefit from this rich inventory of features, which will significantly reduce the necessary amount of training instances. Overlapping representations of complex facts give rise to mapping queries to answers in a Question Answering (QA) system (as e.g. conducted in [200]), which is currently a very active area of research. IBM's Watson system, for example, is a system that works massively parallel by simultaneously using many different approaches to question answering [97], and uses machine learning on high-level features to determine its answers. Since names and values of features are irrelevant in this setting besides their ability to grasp the differentiation granularity needed for the task, annotations generated by the SDM can be easily incorporated and evaluated in such an application-based way. Feature selection mechanisms [cf. 144, inter al.] allow to choose those annotations that correlate best with the task-defined structural regularities.

Secondly, when examining what annotations are the most important ones for solving language processing tasks and tracing back their creation to their data-driven origin, one will find that the procedures of the SDM provide insights into the system of constituents of natural language and could play the role of de Saussure's desperately sought realities as stated in the epigraph of this chapter.

Elaborating on the example of Question Answering, a complete Structure Discovery version of a QA system is now briefly sketched. The task of a QA system is defined by question-answer pairs that have been constructed manually. The challenge in QA is to connect questions to text passages and extract the answer. This is hampered by the fact that, in most cases, the wording of the questions does not exactly match any of the available text passages. In traditional QA, text passages are preprocessed and indexed in a way that fuzzy matching is supported, e.g. by synonym expansion and grammatical canonicalisation. Further, grammatical patterns and transformations are applied to e.g. transform a question like "Who invented the light bulb?" into patterns like "X invented the light bulb" or "X, the inventor of the light bulb" that can be matched on text passages. The linguistic structures, on which these transformation and matching modules operate, are important in intermediate representations, but do not play a direct role in the answer: it is not important to know that e.g. X is in the subject position of "invented", as long as X can be resolved

as the answer to the question. Thus, preprocessing could be replaced by Structure Discovery processes that would add annotations similar to morphology, parts of speech, dependency parses and lexical substitutions to both the text passage and the question. From a multitude of possible realisations and parameterisations of components realising these structural annotations, those are selected that contribute the most in training a system on question answer pairs. For different question types, as e.g. found by clustering on question structure or answer category, subsystems with different components could be learned. Some of the processes will be shared, such as e.g. the morphology component. Others might differ considerably, e.g. domain-specific semantic word classes.

1.5 Connecting Structure Discovery to NLP tasks

Obviously, the iterative building of a stack of Structure Discovery procedures has to be guided by a target application to determine the usefulness of SD annotations. But as opposed to linguistic preprocessing steps, where system behaviour is defined by carefully assigned linguistic annotations by expert annotators, a more task-driven and usage-driven approach to data acquisition is proposed in this section.

The acquisition bottleneck of knowledge processing stems from the fact that it is expensive to create large amounts of high quality data for training or validating automatic systems. There are two ways to widen this bottleneck: either by employing algorithms that better abstract over the data so that less guidance in the form of annotations is needed for similar performance, or by finding a cheaper way of creating the data.

Structure Discovery algorithms perform abstractions and help to minimise the amount of data collection for intermediate processing steps — in the ideal case, the dependency of manual data creation for preprocessing would be removed altogether. For the last step that actually performs the task defined by the application, it might as well be that two sources of data acquisition are suitable, which have not yet been used widely in the NLP community: crowdsourcing and user logs.

Crowdsourcing is a means to distribute work to a potentially large group of workers. On a crowdsourcing platform like Amazon Mechanical Turk⁵ or Crowdflower⁶, requesters define tasks and farm them out to workers. Compared to professional annotators, these workers are comparatively cheap and produce noisier data sets since quality control is more difficult than with expert annotators. Crowdsourcing recently gained increased attention for creating language processing data sets, see [52] for a survey. Criticism of crowdsourcing linguistic datasets includes, amongst other things, that crowd workers are not linguists and therefore miss subtle distinctions and are not able to produce good enough data for linguistic preprocessing steps.

⁵ www.mturk.com [August 31, 2011]

⁶ <http://crowdflower.com> [August 31, 2011]

While it seems not feasible to create e.g. a treebank with crowdsourcing, since workers cannot be sufficiently trained to produce consistent annotations according to a linguistic theory, the picture changes when shifting the preprocessing to Structure Discovery processes and letting untrained workers do what they are good at: be users of a system.

An advanced system of natural language understanding should learn from the users it is operated by — in analogy to today’s search engines, that are heavily tuned by mining logs of user behaviour. For example, a spelling correction system could memorise what corrections were accepted by the user in what contexts and use this data for adjusting its model. As this sort of feedback is very indirect and noisy, a large amount of user data is needed to effectively tune such a system. Also, the system needs a certain quality to begin with — users will only operate the system regularly to produce enough usage data if they have a benefit from doing so. In order to reach this quality threshold, the benefit can be provided by paying users via crowdsourcing, which plays the role of a segue between the content-data-driven Structure Discovery layer on the one hand, and user-data-driven systems on the other hand.

1.6 Contents of this Book

This section provides a guideline for reading this book. Depending on whether the reader is more interested in the theoretical underpinnings or the more practical aspects of this work, different parts should be in focus. Chapters 2, 3 and large parts of Chapter 4 are of theoretical nature. These parts are necessary to understand the paths taken in the more application-oriented Chapters 5, 6 and 7, but can probably be merely used for reference if the reader is less interested in constructing her/his own Structure Discovery processes, but rather in using existing SD implementations.

1.6.1 *Theoretical Aspects of Structure Discovery*

Readers familiar with basic notions of graph theory can safely skip Section 2.1 and merely use it for reference to the exact definitions of notions used in later chapters. Section 2.2 is dedicated to scale-freeness and the Small World property of networks, describing a number of emergent random graph generation models. In Sections 3.1 and 3.2, we look at quantitative characteristics of language data. The notion of the word co-occurrence graph, which reappears frequently in later chapters, is defined in Section 3.2.1. In Section 3.3, an emergent random text model is defined, which models quantitative characteristics of language much more closely than other graph generation models or random text models. This model contributes to the understanding of the origin and the emergence of language structure, but is not directly related to applications described later. Section 4.1 on clustering methods in general and

graph clustering methods in particular discusses reasons for the need of a fast graph clustering algorithm, but might not contain a lot of new material for readers familiar with these fields. Understanding the definition of the Chinese Whispers algorithm in Section 4.2, however, will be very helpful for the applied part of the book.

1.6.2 Applications of Structure Discovery

Chapter 5 describes an SD system for language separation. This method is not only able to find small injections of foreign language into monolingual corpora, but also to robustly recognise a large number of languages in highly multilingual data, even for non-standard texts such as Twitter data. The reader is pointed to an implementation of the system. Unsupervised part-of-speech tagging is discussed in high detail in Chapter 6. The utility of using unsupervised word class labels is demonstrated on a number of standard natural language processing tasks. Again, an implementation of the system is available for download, as well as a number of already induced taggers for various languages that can be easily incorporated in NLP systems. The topic of Chapter 7 revolves around word senses and meaning. In this chapter, the task is defined through lexical substitution data collected by crowdsourcing, which is also available for download. Unsupervised word sense induction features improve the performance of a supervised lexical substitution system. Also, the selection process for unsupervised features stemming from different parameterizations of the same SD process is demonstrated.

1.6.3 The Future of Structure Discovery

The final chapter summarises the achievements contained in this volume, and gives an outlook on possible directions of future work in Structure Discovery. With each unsupervised and knowledge-free method added to the inventory of Structure Discovery processes, another step is taken towards a data-driven and application-driven approach to natural language processing, where tight control of the preprocessing pipeline is given up in favour of language-independence and domain-independence.



<http://www.springer.com/978-3-642-25922-7>

Structure Discovery in Natural Language

Biemann, C.

2012, XX, 180 p., Hardcover

ISBN: 978-3-642-25922-7