2
Text Chunking

The procedure of segmenting words syntactically in a non-overlapping manner was first characterized as an intuition about how human beings parse a sentence being read. That intuition set the first notion of chunking, and paved the way for further research about its importance and how chunks can help understanding other questions concerning the syntactic structure of sentences.

The idea of tackling text chunking as a Machine Learning problem did not take long to be proposed. At first, approaches targeted simpler chunking schemes, e.g., only roughly classifying some word sequences as noun phrase chunks or verb phrase chunks. It was not until the CoNLL-2000 shared task that a corpus with a complete set of chunk types was introduced. Since then, many effective Machine Learning approaches have been presented.

Text chunking is part of a broader concept called shallow parsing, which includes other tasks whose objective is to recover only a limited amount of information from natural language sentences (30). Some facts can explain why shallow parsing in general, and the text chunking task in particular, are considered relevant. First, not all applications require a complete syntactic analysis, and often a full parse provides more information than needed. One example is Information Retrieval, for which it may be enough to find simple noun phrases and verb phrases (30). Text chunking usually provides enough syntactic information for several such applications. Second, it can serve as a preprocessing step to full parsing itself (5, 57). Also, Wu et al. (71) argue that text chunking also provides an important knowledge representation fundamental, giving the example of a Machine Learning-based system adopted to help measure syntactic development in children, which yields results close to human judgment (53).

In addition, several Machine Learning approaches to NLP problems effectively use text chunking results as a feature. These problems include clause identification (16, 23), dependency parsing (10, 17), semantic role labeling (14, 15, 55) and machine translation (69).

In this chapter, we formally define the text chunking problem and the motivation behind its inception. We also describe some of the approaches

created to solve it, giving special attention to the CoNLL-2000 shared task, which highlighted the applicability of Machine Learning for that purpose. Our analysis of the state-of-the-art for this task is based on the corpus provided for the CoNLL task. Finally, we give more information about the progress of research on tasks related to text chunking and focusing on the Portuguese language.

2.1 Definition and initial motivation

The NLP task of text chunking is defined as dividing a text into "phrases" in such a way that syntactically related words become members of the same phrase. These phrases are non-overlapping, which means that one word can only be a member of one of them at most (60). A better term for such a group of words is "phrase chunk", or rather, just chunk, so to establish a distinction between it and an actual linguistically defined phrase.

Although chunks and actual phrases are similar in function, the definition of a phrase allows recursiveness, meaning that a phrase may be embedded in another phrase. Figure 2.1 shows the syntactic parse tree of a sentence, evidencing its phrase structure, while figure 2.2 displays the resulting chunks for the same sentence.

This definition for chunks was first stated by Abney (5), as a way to provide an initial syntactic foundation to the subsequent full parsing of sentences. The basic inspiration for them comes from the fact that humans seemingly read sentences one "chunk" at a time, since the places where stresses and pauses happen during reading roughly correspond to the insides of chunks and boundaries between chunks, respectively.

Abney argues that psychological confirmation for the existence of chunks comes from the experimental work of Gee and Grosjean (29), who identified patterns of word clustering while observing reading behavior. They called the clusters *performance structures* at the time, and suggested that these could be predicted by syntactic structures called *ϕ-phrases*.

By their definition, ϕ-phrases are generated when the sentence is broken after each syntactic head that is a content word. Abney takes advantage of this definition to establish how chunks behave. However, in order to compensate for some shortcomings of ϕ-phrases, he gives a syntactic structure to chunks consisting of a connected subgraph of the sentence's parse tree, and decides they should be based around the concept of *major heads* instead. Major heads are all content words excluding the ones that appear between a function word and the content word that this function word selects.

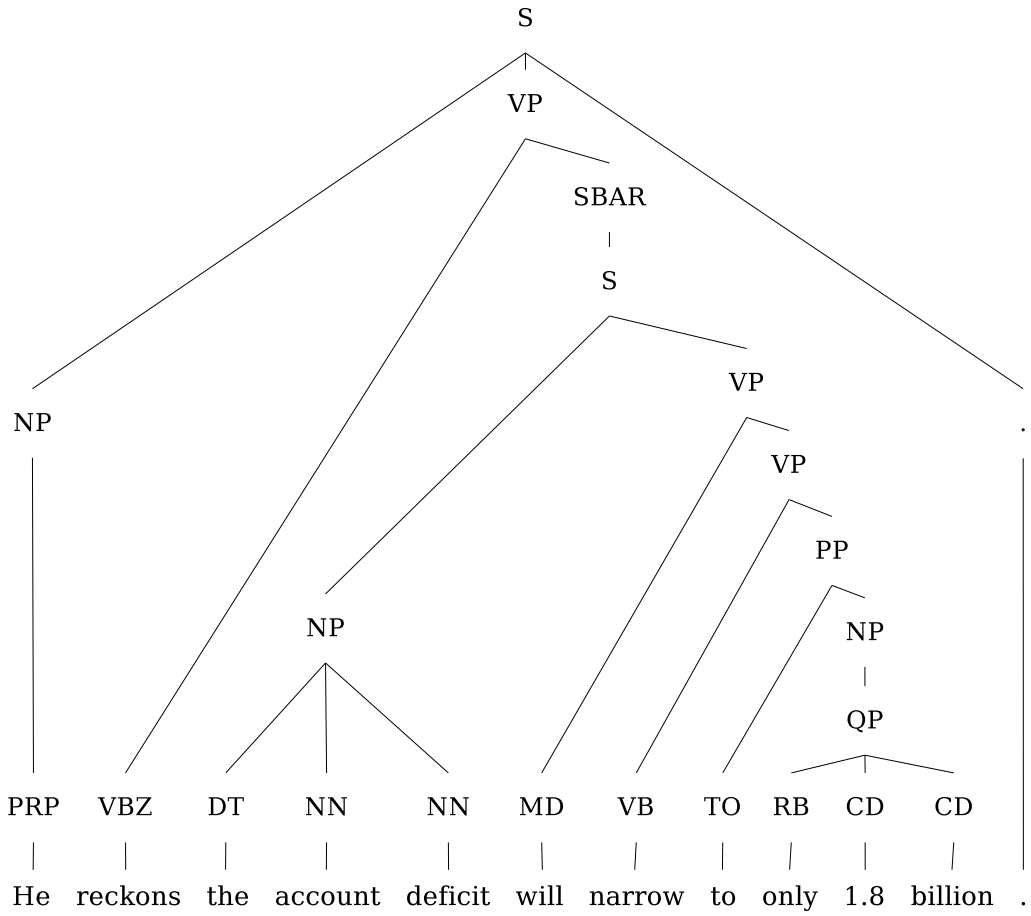

Figure 2.1: Example of a parse tree

$[_{NP}$ He] $[_{VP}$ reckons] $[_{NP}$ the account deficit]
$[_{VP}$ will narrow] $[_{PP}$ to] $[_{NP}$ only 1.8 billion] .

Legend
NP: noun phrase
VP: verb phrase
PP: prepositional phrase

Figure 2.2: A sentence split into phrase chunks

During [$_{NP}$ the third quarter] , [$_{NP}$ Compaq] purchased
[$_{NP}$ a former Wang Laboratories manufacturing facility]
in [$_{NP}$ Sterling] , [$_{NP}$ Scotland] , which will be used for
[$_{NP}$ international service and repair operations] .

Figure 2.3: Ramshaw and Marcus' baseNP structure

Abney then proceeds to propose a chunking parser, whose first step is the generation of word clusters by a chunker, which are then linked together by the addition of missing syntactic arcs by an attacher. His idea was that chunking was reasonably solvable by a finite state method, and that the higher-level decision of how chunks are to be combined should be postponed. Thus, his implementation for this chunker consists in a non-deterministic version of an LR parser (35), using a hand-built grammar.

2.2 Introduction of Machine Learning methods

The work that first inspired research on the application of Machine Learning techniques to the chunking task is the one of Ramshaw and Marcus (51). Using the Transformation-Based Learning (TBL) mechanism, proposed by Eric Brill (9), they create automatic chunk extractors using an English corpus of Wall Street Journal text from the Penn Treebank (41) as training data.

Chunk data for the used corpus is algorithmically derived. Ramshaw and Marcus work with two distinct types of chunk structure. The first is what they refer to as "baseNP" chunks, and the rationale behind them is the identification of the initial portions of non-recursive noun phrases up to the head, including determiners but excluding post-modifying prepositional phrases or clauses. These are extracted by selecting noun phrases (NPs) that contain no other nested NPs. Figure 2.3 displays a sentence and its corresponding baseNP chunks. The second type consists of noun phrase and verb phrase (VP) chunks, structured in a manner more similar to Abney's original proposal, but with a caveat: both NP and VP kinds include some elements that are not nominal or verbal, respectively. Chunks classified as NP also include prepositions, for instance, and every other chunk is classified as VP. We demonstrate this structure type in figure 2.4.

Their work also set another milestone for text chunking research: treating it as a classification problem. Ramshaw and Marcus decided that, to be able to apply the TBL technique, every token in the corpus would have related tags corresponding to their POS tags and chunk types. This way, determining

$[_{NP}$ Indexing] $[_{NP}$ for the most part]
$[_{VP}$ has involved simply buying] $[_{VP}$ and then holding]
$[_{NP}$ stocks] $[_{NP}$ in the correct mix]
$[_{VP}$ to mirror] $[_{NP}$ a stock market barometer] .

Figure 2.4: Ramshaw and Marcus' partitioning chunk structure (NPs and VPs)

the chunk structure of a sentence is a matter of classifying every word with its appropriate chunk tag, which encodes the chunk to which the given word belongs.

Ramshaw and Marcus' approach, both in the matter of treating chunking as a classification problem and in that of using a TBL-related technique, is very similar to the present work. Therefore, we postpone the discussion about details of the classification approach and the Machine Learning method to following chapters.

2.3 A complete chunk definition and the CoNLL-2000 shared task

Although the work of Ramshaw and Marcus introduced the identification of distinct kinds of chunk, most subsequent research in the area focused on identifying NP chunks (66, 12, 47).

A problem similar to NP chunking, but slightly more elaborate, was established as the first shared task proposed for the Conference on Computational Natural Language Learning, in 1999 (1). The idea of a shared task, repeated every year since then with a different theme, is to allow participants to compare their research advances by testing their learning systems on the same training and test corpora.

The shared task at that occasion, NP bracketing, consisted in recognizing all noun phrase structures in a text. For that purpose, the application of the same corpus used by Ramshaw and Marcus in the aforementioned work was suggested.

Nevertheless, at the same time, some researchers started including other chunk types in their investigation of data-driven methods for shallow parsing. For instance, Veenstra (65) includes prepositional phrase (PP) chunks in his extractor, and Buchholz et al. (11) also consider adjectival phrase (ADJP) and adverbial phrase (ADVP) chunks in their work.

A comprehensive definition for chunks in the English language was finally established for the CoNLL-2000 shared task (60). As in Ramshaw and Marcus' work, it also uses Wall Street Journal sections from the Penn Treebank, and chunks are extracted in a programmatic manner. The derived types are based

on the syntactic category part of the bracket labels in the Treebank, and, generally speaking, chunks contain everything to the left of and including the syntactic head of the constituent of the same name. In this case, many other chunk types are taken into account, though it is important to notice that NP, VP and PP chunks account for about 95% of all chunks. Table 2.1 shows the list of chunk types present in the resulting corpus. Moreover, POS tags generated by the system developed by Brill (8) are also provided in this corpus.

Chunk type	**Description**
ADJP	adjectival phrase
ADVP	adverbial phrase
CONJP	conjunction phrase
INTJ	interjection
LST	list marker
NP	noun phrase
PP	prepositional phrase
PRT	particle
SBAR	relative or subordinate clause
UCP	unlike coordinated phrase
VP	verb phrase

Table 2.1: Chunk types in the CoNLL-2000 corpus

The results presented by the participant systems at CoNLL-2000, along with some extra results added later by the CoNLL organization to showcase the evolution of the state-of-the-art, are displayed in table 2.2. The table also shows the results corresponding to the system considered as baseline, that is, a naive system whose results serve as a lower bound benchmark for the task. In this case, the baseline system (BLS) attributes to a given token the chunk tag which was most frequently associated with the token's POS tag in the training data.

The submitted models can be split in four categories: rule-based systems (18, 68, 32), memory-based systems (67), statistical systems (74, 36, 48, 49) and combined systems (37, 64, 56). These ensemble models, which relied on the combination of the results from several learning systems, were the most effective ones at the time.

2.4 Overview of state-of-the-art approaches

The CoNLL-2000 shared task set a new standard to the chunking task, providing incentive for a number of different Machine Learning approaches to be proposed and allowing for the verification of which ones seemed better suited to solve it. In the following years, refinements of those ideas were presented,

Participant[1]	**Precision (%)**	**Recall (%)**	$F_{\beta=1}$
* **Zhang et al.** (72)	94.29	94.01	94.13
* **Kudo and Matsumoto** (38)	93.89	93.92	93.91
* **Carreras and Màrquez** (13)	94.19	93.29	93.74
Kudo and Matsumoto (37)	93.45	93.51	93.48
van Halteren (64)	93.13	93.51	93.32
Tjong Kim Sang (56)	94.04	91.00	92.50
Zhou et al. (74)	91.99	92.25	92.12
Déjean (18)	91.87	92.31	92.09
Koeling (36)	92.08	91.86	91.97
Osborne (48)	91.65	92.23	91.94
Veenstra and van den Bosch (67)	91.05	92.03	91.54
Pla et al. (49)	90.63	89.65	90.14
Johansson (32)	86.24	88.25	87.23
Vilain and Day (68)	88.82	82.91	85.76
Baseline system	72.58	82.14	77.07

Table 2.2: $F_{\beta=1}$ score of CoNLL-2000 shared task systems

pushing the state-of-the-art for the task based on the CoNLL English corpus even further in terms of effectiveness and chunking speed. The state-of-the-art for this corpus is currently slightly above an $F_{\beta=1}$ score of 94. The $F_{\beta=1}$ score aggregates both the precision and the recall in the identification of chunks.

Results for some of these subsequent systems are also displayed in table 2.2. Techniques applied by these systems include Hidden Markov Models (HMM), Support Vector Machines (SVM) and a generalization of the Winnow algorithm. We describe some of the most successful approaches in the next subsections, emphasizing their design decisions.

2.4.1 Winnow algorithm-based approach

Zhang et al. (73) use a variation of the Winnow algorithm, which they call *regularized Winnow*, to approach the chunking problem.

The basic Winnow algorithm learns a linear classifier based on training data. The aforementioned authors modify it in order to guarantee that the algorithm will converge even when using non-linearly non-separable features, something that the original version may not do.

This change allows for the use of a large number of features. For example, assuming that $tok_{-c}, tok_{-c+1}, ..., tok_0, ..., toc_{c-1}, tok_c$ is a string of tokenized text, the POS tag for tok_i is pos_i and the chunk tag for tok_0 is wanted, the following set of features is used for a fixed c:

[1]Scores marked with * were published after the end of the conference.

– first order features: tok_i and pos_i $(i = -c, ..., c)$
– second order features: $pos_i \times pos_j$ $(i, j = -c, ..., c; i < j)$ and $pos_i \times tok_j$ $(i = -c, ..., c; j = -1, 0, 1)$

Because this approach is a sequential process, meaning that chunk tags are determined token by token in a sentence, the already predicted tags can also be used as features when detecting the tag for a subsequent token.

This Winnow algorithm-based model is one of the best performing among non-ensemble methods, i.e., the ones that do not combine the results of many classifiers.

2.4.2 Support Vector Machines approach

SVMs are employed by Kudo et al. (38). Since an SVM is a binary classifier, it needs to be extended to a multi-class classifier to capture a big range of chunk types. In order to achieve that, they take advantage of *pairwise classification*: to identify K classes, $K \times (K-1)/2$ classifiers are built, each of them taking a pair of classes into account, and the final decision is established by their weighted voting.

As with the Winnow-based system, features considered by this model include the ones given by the surrounding context. This means that, when classifying a certain token, tags of neighbor tokens are used as features. The authors also experiment with the weighted voting settings, distributing the voting weight between classifiers according to four distinct methods.

The first incarnation of this SVM model had the best overall results at the time of the CoNLL-2000 shared task. Other authors use similar SVM approaches to improve upon this result. Wu et al. (70), for instance, apply a method to increment the feature set for unknown words, that is, words not seen during training that appear at extraction time.

2.4.3 Hidden Markov Model approach

A system created by Molina et al. (46) is based on HMMs, statistical models that treat the task at hand as a Markov process with hidden states to be discovered. In a typical setting, these states represent the possible output classes for the task.

This approach differs from the previously discussed ones in that it is not classification-based, being a generative model instead. Its goal is to reveal the most likely sequence of output chunk tags for a given input sentence in terms of probability. Because the probability of the whole output sequence

is maximized, therefore not just the likelihood for individual chunk tags, the complete set of input data influences the result tags for all tokens.

It is argued by the authors that using the plain input and output tag sets leads to poor results when using this HMM approach. In the case of input data, the whole set of possible words and POS tags makes the input vocabulary too large, resulting in poor estimations. As for the output tag set, the application of too few and generic classes, like the typical set of chunk tags, yields inaccurate models.

These issues are solved by applying a *specialization function* that transforms both the input and the output tag sets. As an example, we consider the training tuple $\langle w_i \cdot p_i, ch_i \rangle$, which denotes that $w_i \cdot p_i$, the combination of all possible words and POS tags, is defined as the input vocabulary, and the output set is simply ch_i, the set of chunk tags. A possible specialization function is the following:

$$f_s(\langle w_i \cdot p_i, ch_i \rangle) = \langle p_i, p_i \cdot ch_i \rangle$$

The result of this function are training tuples in which only POS tags serve as input, and in which output tags are complemented with the POS tag associated with each input word.

Molina et al. experiment with a number of specialization functions involving POS tags, chunk tags and selections of words. Their system eventually demonstrates results competitive with the best ensemble methods.

2.4.4 Other approaches

A number of other theoretical frameworks has been used to tackle the text chunking task during and after the CoNLL shared task. Among those, we can cite voted perceptrons (13), memory-based learning (59), maximum entropy (36), Conditional Random Fields (54), and also ETL (42).

Another issue considered by some authors is that some of the state-of-the-art systems, though demonstrating very high performance, would not be satisfactory under real use conditions because of their low extraction speed. Wu et al. (71) present a work where they specifically deal with this issue, implementing a substantially faster system while not sacrificing much of the overall performance.

2.5 Chunking for Portuguese corpora

The majority of recent research on text chunking uses the English CoNLL corpus to benchmark their results. In general, Machine Learning approaches to NLP problems are known to be noticeably language independent, meaning that models that work well with one language are expected to have good performance when applied to another language. Nevertheless, each language has particular peculiarities that call for specific design decisions. Thus, it is reasonable to investigate solutions to processing problems for languages individually.

The Portuguese language has expectedly not received the same amount of attention concerning the text chunking task as English. As far as we can determine, recent research in the area for the language targets mainly the identification of noun phrases. We summarize some of the most notable advances involving Machine Learning techniques.

Dos Santos (20) does a thorough study about the application of TBL for noun phrase extraction in Portuguese. Freitas et al. (27) propose a specific definition for NPs in Portuguese called *reduced noun phrases*, and take advantage of the TBL framework set by dos Santos to provide an extractor for those phrases. Milidiú et al. combine TBL and HMM with semi-supervised techniques (43), and later adopt the ETL algorithm in order to improve upon the results obtained by Freitas et al. (42). This last approach consists in a committee composed of three ETL classifiers with different window parameters. Finally, dos Santos (21) reports even better results for the same dataset using an ETL commitee.

We give more details about TBL and ETL in chapter 4. In table 2.3, we show the results obtained for the reduced noun phrase corpus established by Freitas et al. (27), called SNR-CLIC.

NP extractor	**Precision (%)**	**Recall (%)**	$F_{\beta=1}$
Freitas et al. (27)	83.80	84.20	84.00
Milidiú et al. (42)	88.62	89.67	89.14
dos Santos (21)	89.66	89.51	89.58

Table 2.3: Past results for Portuguese noun phrase extraction

Chunks in the Portuguese language have been used in a practical context by at least one system. The CoGrOO project (34), a grammar checker for the OpenOffice software suite[2], has a chunk module capable of extracting NPs and VPs. This module is based on the Maxent package of the Apache OpenNLP

[2] `http://www.openoffice.org`

open source project[3], which aggregates maximum entropy models for various NLP tasks.

Since the time of CoGrOO's implementation, the OpenNLP project has refined its efforts on its supported tasks. In chapter 5, we compare our proposed chunk extractors with the one from OpenNLP in terms of performance, employing the same train and test corpora.

[3] `http://incubator.apache.org/opennlp`