3
A Token Classification Approach

For the past decade Supervised Machine Learning algorithms have been applied to Natural Language Processing problems with great success. Being strongly language independent, approaches that achieve good results in one language can be applied to other languages in a almost *out-of-the-box* way. Generally, the only requirement is the existence of annotated *corpora* in the other language.

Among all the Machine Learning approaches used in NLP, the *token classification* is definitely the most used. In token classification, the first step is to tokenize the given input text, i.e., simply to identify the start and end of the words and punctuation in the text. Then, the task is to assign to each token the *class* (or *tag*) that it belongs to. It is exactly in this selection of classes that lies the solution to the task in question.

For problems like Part-of-Speech tagging, the selection of classes is straightforward: it is exactly the part-of-speech classes that each word might have – that is, noun, verb, adjective, pronoun, preposition, etc. Other tasks, like Chunking, require subtle changes when creating their classes, since one or more adjacent tokens might be part of the same *type* of phrasal chunk, but actually be part of different chunks. In this case, as well as in Named Entity Recognition, tagging styles such as IOB [76] and IOB-2 [4] are used to avoid classification ambiguity.

Tasks like Clause Identification or Semantic Role Labeling require the identification of the scope of a clause or an argument. In these tasks, different tags are used to identify the start and the end of a scope or if the token is not part of any of those. In all those examples, once a tagset has been proposed, all the range of Machine Learning classification algorithms can be promptly applied and evaluated. Also, different tagsets can achieve different results for the same task and *corpus*, since different classes lead to different generalizations of Machine Learning models. Therefore, we need an appropriate set of classes for each problem.

In the first section of this chapter, we propose and describe a tagging style

that allows the dependency parsing to be solved by a token classification approach, as well as some statistics that show the adherence of our set of classes to the dependency parsing problem. Then, we describe how the proposed set of classes can be used to solve the dependency parsing and can also be decomposed into subtasks that can be solved independently. Further, we present how the proposed set of classes allows the creation of a statistically built baseline classifier for dependency parsing. Finally, we describe some fundamental features for dependency parsing and how those can be used to create derived features, which can improve parser's accuracy.

3.1 Token Classification Classes

The case of Dependency Parsing, proposes a particular challenge for a *token classification* approach. Take for instance an excerpt from a Portuguese *corpus* as shown in Table 3.1. In this example, the left half of the table shows a sentence in a tabular format. The first column is just a identifier of the position of the token in the sentence, while the second column contains the word forms of the tokens. The third column identifies the head of each token by its position as presented in the first column. So, the head of the first token, *É*, is the ninth token, *tem*; the head of *isso* is *por*; and *diz* is the root of the sentence – represented by a head of position 0.

3.1.1 Absolute Head Position

A simple way of solving this problem as a token classification one is to treat each head position as a class, i.e., creating classes for each possible position in a sentence: *class 1* for tokens whose head is the first token of the sentence, *class 2* when the head is the second token and so on. However, when applying a Machine Learning classification algorithm, such approach would lead to poor generalization of its models.

This issue is explicit in the right half of Table 3.1. Now, the same sentence is presented with a slight difference: a token *ele* is added at the sixth position, increasing the position identifier of all tokens that come after it. Even though only one token was added and syntactically its impact was almost none, this small difference changes the class of eleven of the thirteen tokens in the sentence. Since even similar sentences can have such different classifications for each token, classification algorithms would not be capable of making good generalizations

Id	Word	Head		Id	Word	Head
1	É	9		1	É	**10**
2	por	9		2	por	**10**
3	isso	2		3	isso	2
4	que	9		4	que	**10**
5	,	6		5	,	**7**
			$\Longrightarrow$	6	**ele**	
6	diz	0		7	diz	0
7	,	6		8	,	**7**
8	não	9		9	não	**10**
9	tem	6		10	tem	**7**
10	pena	9		11	pena	**10**
11	de	10		12	de	**11**
12	Bill	11		13	Bill	**12**
13	.	6		14	.	**7**

Table 3.1: Example of Absolute Head Position Classes.

for their models. Therefore, a different set of classes is needed if a classification approach is to be applied to dependency parsing.

3.1.2 Head Displacement

As a head position tagging style lacks the capability of making good generalizations, a better set of classes is needed. A immediate improvement is to use the *displacement* between the token and its head as its class. In this case, instead of using the head position in the sentence to locate it, we use the relative position from the token to its head. This is shown in table 3.2, where the same example is presented with head displacement classes.

In this case, only addition or removal of tokens between a token and its head changes its class. Furthermore, these classes are sentence length independent, since the absolute position of the head is not used, but a relative to the token distance. However, in the simple example above a slight change in the sentence still changes the class of four of the thirteen tokens, what shows how this set of classes leads to poor generalization of models.

3.1.3 Part-of-Speech Head Displacement

The dependency parsing can be seen as a task where for every token in a sentence another token, its head, must be identified. Hence, a set of classes for

Id	Word	Head		Id	Word	Head
1	É	+8		1	É	**+9**
2	por	+7		2	por	**+8**
3	isso	-1		3	isso	-1
4	que	+5		4	que	**+6**
5	,	+1		5	,	**+2**
			$\Longrightarrow$	6	**ele**	
6	diz	0		7	diz	0
7	,	-1		8	,	-1
8	não	+1		9	não	+1
9	tem	-3		10	tem	-3
10	pena	-1		11	pena	-1
11	de	-1		12	de	-1
12	Bill	-1		13	Bill	-1
13	.	-7		14	.	-7

Table 3.2: Example of Head Displacement Classes.

this problem must be able of identifying any token in a sentence. Additionally, as presented before, this set of classes must be robust to small changes so Machine Learning algorithms can accurately generalize the given *corpus*.

In our tagset, we join three pieces of information about the head of the token to pinpoint it in the sentence. The first is whether the token's head comes before (*left*) or after (*right*) the token. The second identifies the type of the token's head. Any available information that discriminates different words and is available in the *corpus* can be used. In this work we use part-of-speech classes to identify the type of the token's head. Finally, the third is a *distance counter* of how many tokens there are between the token and its head *with same type of the head*. If the token is the root of the sentence, then a special tag *root* is used.

Table 3.3 shows the previous Portuguese example, now with the new tagset in the right column. The tag is composed by joining the *distance counter*, the *part-of-speech of the head* and if the head comes before (*left*) or after (*right*) the token. Therefore, *1_v_R* stands for *First verb to the right*, *2_v_L* stands for *Second verb to the left* and *1_prp_L* stands for *First preposition to the left*,

Since we do not use the exact position of the token to identify it, but instead use part-of-speech as a way of finding another token, our tagset is a lot more robust to changes than the previous tagset. This can be seen in Table 3.4, where we apply the same modification as before.

In this example, since the added word was a *pronoun* – a part-of-speech with no child token in this sentence – *no changes* happen to our tagset. Actually, only changes between the token and its head that involve a word whose part-of-speech is the same as the token is will change its class, what indicates that our

Id	Word	Part-of-Speech	Head	Special Tagset
1	É	adv	9	2_v_R
2	por	prp	9	2_v_R
3	isso	pron	2	1_prp_L
4	que	adv	9	2_v_R
5	,	punc	6	1_v_R
6	diz	v	0	root
7	,	punc	6	1_v_L
8	não	adv	9	1_v_R
9	tem	v	6	1_v_L
10	pena	n	9	1_v_L
11	de	prp	10	1_n_L
12	Bill	prop	11	1_prp_L
13	.	punc	6	2_v_L

Table 3.3: Example of Part-of-Speech Head Displacement Classes.

Id	Word	Head
1	É	2_v_R
2	por	2_v_R
3	isso	1_prp_L
4	que	2_v_R
5	,	1_v_R
6	diz	root
7	,	1_v_L
8	não	1_v_R
9	tem	1_v_L
10	pena	1_v_L
11	de	1_n_L
12	Bill	1_prp_L
13	.	2_v_L

$\Longrightarrow$

Id	Word	Cpos	Head
1	É	adv	2_v_R
2	por	prp	2_v_R
3	isso	pron	1_prp_L
4	que	adv	2_v_R
5	,	punc	1_v_R
6	**ele**	pron	
7	diz	v	root
8	,	punc	1_v_L
9	não	adv	1_v_R
10	tem	v	1_v_L
11	pena	n	1_v_L
12	de	prp	1_n_L
13	Bill	prop	1_prp_L
14	.	punc	2_v_L

Table 3.4: Changes with Part-of-Speech Head Displacement Classes.

tagging style is more robust and allows for a better generalization of Machine Learning models.

3.1.4 Model Statistics

Supervised Learning algorithms need a large number of examples from a problem domain to accurately generalize their models from the given data. Since the size of our *corpora* is fixed, a model that proposes a high number of classes would lead to fewer examples per class, thus having a negative impact in our

algorithms accuracy.

When analyzing the *theoretical* number of classes according to the model proposed in this work, we are faced with a prohibitive number. Considering a set of 15 part-of-speech tags, a set of 20 possible *distances* and two sides to locate a token's head, we would have 600 possible classes for each token.

However, only a fraction of those classes is needed since most of those, albeit possible, simply do not occur in actual sentences. This can be noticed when evaluating our model in the *corpora* made available by the occasion of the CoNLL 2006 shared task, further described in Chapter 5. Table 3.5 shows how many classes are needed to represent every token's head in the training set when using the *coarse-grained part-of-speech* to identify the head's type.

Furthermore, even a smaller set of classes is sufficient to cover most tokens in the *corpora*. This is noticed in Table 3.5 right columns, where it shows how much of the *corpora* is covered with the top 5, top 10 and top 20 most common classes. For these languages, more than 95% of the token's heads can be identified with just 20 classes.

Language	**Total number of Classes**	**Top 5 coverage**	**Top 10 coverage**	**Top 20 coverage**
Danish	118	66.3%	88.4%	95.7%
Dutch	105	70.1%	89.0%	97.2%
Portuguese	135	75.1%	89.0%	96.1%

Table 3.5: Class Coverage for the Training Set.

However, this number increases when using *fine-grained part-of-speech* to identify the head's type, hence, *coarse-grained part-of-speech* is used in further experiments. Table 3.6 presents the total number of classes needed to cover the training set when using each type of part-of-speech.

Language	**Danish**	**Dutch**	**Portuguese**
Coarse-grained	118	105	135
Fine-grained	164	416	150

Table 3.6: Number of Classes by Part-of-Speech Granularity.

When analyzing the statistical distribution of our tagset, we noticed that, although there are tags where the distance information is greater than 2, they are statistically rare. Table 3.7 presents each *corpus* coverage according to the *distance* value in our tagset. Although when only considering *root* distance it covers less than 10%, adding the tokens that have *distance* of *one* to the head covers more than 88% of the *corpora*. With distances of up to 4, more than 98% of the *corpora* is covered. This shows that the part-of-speech of the head is a

strong information in dependency parsing, indicating the adherence of our set of classes to this problem.

	Only Root (%)	Up to 1 (%)	Up to 2 (%)	Up to 3 (%)	Up to 4 (%)
Danish	5.56	88.98	95.55	97.79	98.83
Dutch	8.21	92.96	98.56	99.62	99.89
Portuguese	4.39	88.20	95.14	97.54	98.71

Table 3.7: Coverage of each Distance Value.

3.2 Task Decomposition

With the proposed Token Identifier tagging style the Dependency Parsing can be solved by applying any classification algorithm in a token classification approach. One way of modeling this is as a *one task only* where the classification algorithm directly maps from the input features to the proposed classes.

Moreover, since our tags use three information to find the token's head, we can split the dependency parsing problem into three analogous subtasks. The first subtask is to identify if the head of the token comes before (left) or after (right) the token. The second subtask is to identify the part-of-speech tag of the token's head. A third subtask is to find at what distance from the token is its head, counting only the tokens with the same part-of-speech as the head. In all these subtasks, there is a *root* class when the token is root of the dependency tree.

Breaking the parsing down into subtasks allows Machine Learning models to specialize in simpler tasks, therefore improving their accuracy.

Finally, the results of these three subtasks can be joined, therefore solving the dependency parsing. Additionally, a final classification algorithm can be applied, using the joint results as an initial classifier to further improve the parser accuracy.

3.3 Baseline Classifiers

Since in a token classification approach the task consists in correctly predicting a class for each given token, our tagging style also allows us to statistically build a baseline classifier.

For the *one task* approach, we propose the following baseline classifier: assign to each token the most frequently seen class for its part-of-speech. For

instance, in Portuguese, each *pronoun* is classified as *first verb to the right* while each *verb* is classified as *root*. As far as we know, this is the first statistically built baseline system for dependency parsing.

Furthermore, this concept can be extended to the subtasks approach, with the following baseline classifiers. The baseline system for the first subtask classifies each token with the most frequent class for its part-of-speech as seen in the training set. For instance, in Danish, *nouns* are classified as *left* and *interjections* are classified as *root*.

The baseline system for the second subtask classifies each token with the most frequent class for its pair "part-of-speech – side predicted in the previous task". For example, in Portuguese, an *adjective* that has its head side predicted as *left* is classified as *noun*.

Finally, the baseline system for the third subtask also classifies each token with the most frequent class for its pair "part-of-speech – side predicted in the first sub-task". Hence, in Dutch, an *adjective* that has its head side predicted as *left* is classified as *one*.

In Appendix B we present a complete description of our baseline classifiers for each of the three languages used in our experiments.

3.4 Feature Engineering

A common practice to achieve better results in a task is to use information from some previous tasks as input feature to the ML algorithms being used. For Dependency Parsing, part-of-speech and lemma from each token are considered fundamental input features, as well as any other morphosyntactic features. Accordingly, our tagging style makes use of part-of-speech to identify the token's head. Phrase Chunk and Clause provide important shallow-syntactic information, what suggests that they can be used as features to further improve dependency parser's accuracy.

Likewise, one common practice to improve a ML algorithm accuracy, is to create derived features, i.e., features that make explicit an information that is already conveyed in the data set. These features can expose patterns or information that otherwise would be difficult to be recognized by the algorithm.

In this work, we propose and test a great number of derived features. First, counting features are designed to capture sentence complexity. Since verbs are head of subject and object tokens, knowing if there are verbs and how many of them before and after the token helps to indicate the side of its head, hence we use as features *the total number of verbs before and after the token*. Likewise,

the total number of nouns before and after the token might help, since nouns are commonly found as head of adjectives and prepositions. Similarly, *the total number of punctuation tokens before and after the token* is a good clue to clause boundaries, a strong syntactic information.

Also, *the sequence of part-of-speech tags before and after the token* as a categorical feature, can help capture the sentence complexity.

Since each verb has a particular predicate frame, i.e., expected type of subject, as well as, number and type of expected objects and complements, identifying *the lemma of the closest verb before and after a token* can be a strong derived feature for dependency parsing.

Finally, verbs and nouns are the most common heads in dependency graphs and in this work we found that the most common errors are the ones where the classifier mistake the head between those two types. Hence, features that indicate the context of the closest verb and noun to a token might help differentiate between those cases. Therefore, we propose as derived feature the *part-of-speech of the neighbor tokens of the closest verb and noun before and after the token*.