Using Dominance Chains to Detect Annotation Variants in Parsed Corpora

Pablo Faria

Abstract—In this paper, some results on the detection of variation in annotation in parsed corpora or *treebanks* are presented. Treebanks are generally built by means of using both automatic tools (i.e., taggers and parsers) and human intervention. In this process, inconsistencies (and, thus, variation) in the annotation arise, caused by a number of factors, for instance, disagreement in interpretation, incomplete or unclear annotation guidelines, etc. In this study, the algorithm for automatic detection of variation proposed in [1] is evaluated against the Tycho Brahe Corpus (TBC, [2]) and compared to an alternative implementation where variants of annotation are characterized by means of "dominance chains". Experimental results demonstrate that the modified version has better relative precision and recall than the original method.

Keywords—treebank, inconsistency detection, syntactic annotation, dominance chain, computational linguistics

1 Introduction

The last decades of corpus linguistics have seen the construction of large syntactic annotated (i.e., *parsed*) corpora, called *treebanks*, some of them consisting of millions of words. The process of building such large databases is costly and time consuming, taking several years of continuos work. Although this measure is highly variable and depends on a number of factors, an example of this laborious process is the Tycho Brahe Parsed Corpus of Historical Portuguese (TBC): its publicized version ([2]) has taken 14 years to reach the approximate number of 700,000 tokens.

In general, parsed corpora are built by means of both automatic tools (i.e., taggers and parsers) and human intervention: once current state-of-the-art parsers hardly obtain more than 80% of accuracy, syntactic annotation still requires a significant effort from human annotators or reviewers in order to fix parsing errors and omissions. Furthermore, this intervention is generally carried out by a team of reviewers, frequently showing different degrees of ability and experience with syntactic annotation.

It is thus inevitable that inconsistencies in the annotation arise, caused by a number of factors, for instance, disagreement in interpretation, incomplete or unclear annotation guidelines, lack of experience, among others. As a consequence, many teams perform a double-revision strategy where the revisions of less experienced reviewers are subject to a final revision by a more experienced one. Unfortunately, many inconsistencies still survive this two-stage process. Such inconsistencies may affect the accuracy of the parser (see, for instance, [3]), since the training corpus submitted to it is continually increased by the last revised texts.

In this scenario, methods for detecting inconsistencies play a very important role. They provide a fast way of checking the overall consistency of a corpus while providing exact information about the kind of (common) inconsistencies found. With this information in hand, corrections can be directly made to the current corpus, adjustments can be made to annotation guidelines, and the team of reviewers can be trained with special emphasis on the main sources of disagreement or difficulties.

The development of such methods is chal-

• *Post-doctoral fellow*
Universidade Estadual de Campinas (UNICAMP)
E-mail: pablofaria@gmail.com

Thanks to São Paulo Research Foundation – FAPESP – for funding this research (Grant no. 13/18090-6). Also thanks to the reviewers for their comments and suggestions.

lenging and poses significant computational and linguistic questions, such as matters of efficient computation and grammatical covering of the data. On the computational side, we have to find algorithms capable of efficiently processing the corpus and extracting the relevant information, keeping the complexity of the task under control. On the linguistic side, there are different types of inconsistencies which vary in their grammatical complexity: some are just part-of-speech or syntactic labeling errors while others may involve grammatical phenomena for which the correct interpretation/analysis is not at all clear.

Current methods of detection – such as the ones discussed here, but also alternatives in [4] and [5] – show a relative success in detecting simpler inconsistencies but seem to be limited regarding the more complex cases. Moving towards the latter is thus the main goal of this area of research. In this paper I present some results on the detection of variation in annotation in parsed corpora or *treebanks*. The method of detection applied here is a version of the method proposed in [1], modified in order to use "dominance chains" as the basis for determining cases of variation in annotation. Results demonstrate that the modification increases the relative precision and recall of the algorithm.

The paper is organized as follows. **Section 2** introduces some discussion about inconsistencies in parsed corpora. In **Section 3**, the core aspects of the method of detection in [1] and the its modified version are presented and discussed. Experimental results are reported on **Section 4** followed by a discussion. The paper ends with some concluding remarks in **Section 5**.

2 Inconsistencies in a corpus

One of the most important properties of a corpus is its internal consistency, i.e., the extent to which multiple occurrences of equivalent (sequences of) elements are consistently annotated in the same way. Many factors, such as the complexity and clearness of the annotation system, the experience and ability of the team of annotators, the accuracy of the automatic tools employed, among others, are related to and may be worked out to improve the level of consistency in annotation ([6], [7], [8]).

Blaheta ([9]) discusses the most recurrent types of inconsistencies which he classifies into three categories: *detectable* errors ("type A"), *fixable* errors ("type B") and *systematic inconsistencies* ("type C"). Type A errors, as Figure 1[1] shows, are those where the "annotator's *intent*" is still clear, although the markup is wrong due to lack of attention or confusion. In general, Type A errors can be easily fixed since their correct form can be automatically applied for each and all instances of the errors.

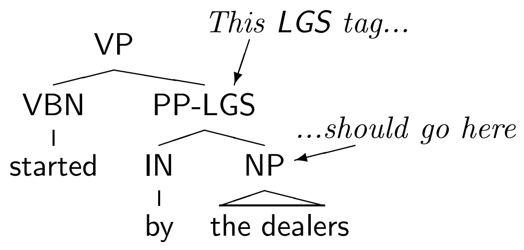
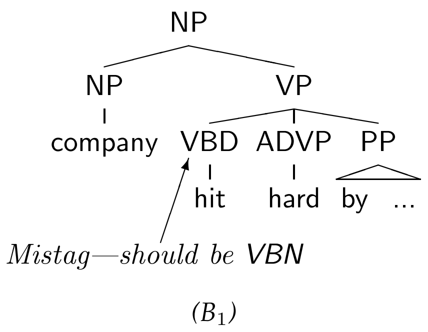


Figure 1. "Type A" error as defined in [9].

Type B and C errors, on the other hand, demand human intervention in order to distinguish errors from true ambiguities, since some variations in annotation are legitimate. The difference between B and C categories is that the corrections to be made to Type B errors, once detected for certain, are clear. Blaheta subdivides Type B errors into two subcategories, B_1 and B_2 exemplified in Figure 2, the former being easier to detect than the latter one.

Type C errors, however, are those to which the annotation guidelines lack orientations, are neutral (i.e., there are guidelines but they depend on the actual interpretation given to the data) or are not clear about. In these cases, the annotators are left with their own intuitions, giving rise to possibly more subtle and complex inconsistencies not only among annotators but also in the annotations made by the same individual. Figure 3 shows an example of a piece of data interpreted in two different ways. Note that both involve the same words and part-of-speech tags but display distinct syntactic phrasing.

1. Figures 1 and 2 are original examples from [9].



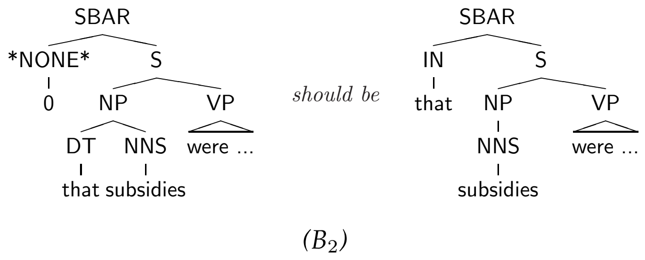


Figure 2. Examples of “Type B” errors ([9]).

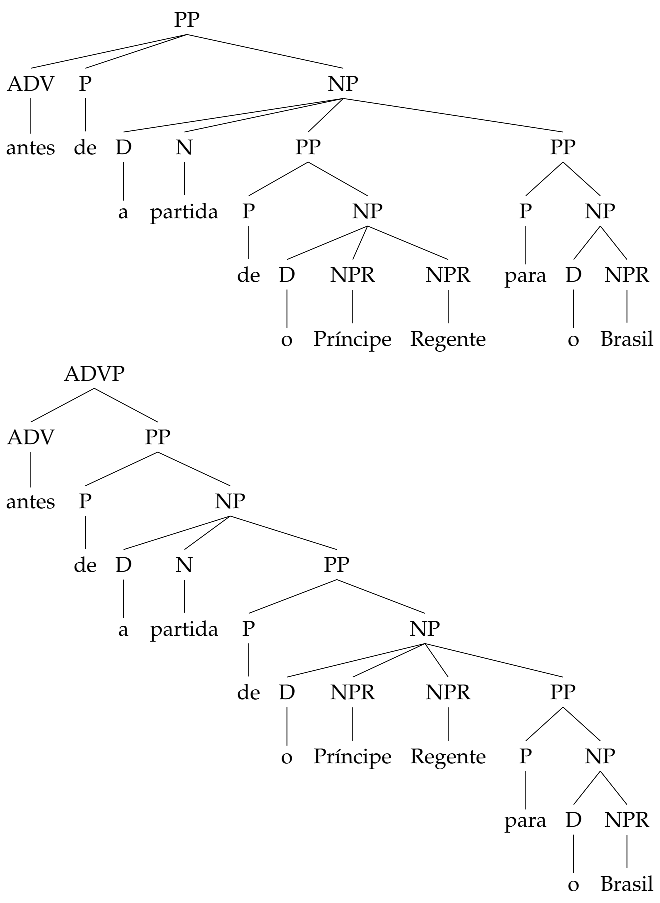


Figure 3. Examples of “Type C” errors in the TBC.

3 Detection of Inconsistencies

In this study, a partial implementation of [1] – which is based on ideas from [10] – was conducted along with an alternative implementation of it in order to assess a particular intuition on the nature of the problem. In this section, first the core aspects of the method in [1] are presented. Next, the alternative method is explained along with the concept of “dominance chain”.

3.1 Finding and distinguishing variation

Dickinson & Meurers ([10], [1]) develop their methods around the notion of *variation*: the possibility that particular tokens or sequence of tokens may be assigned distinct markups (e.g, part-of-speech tags and/or syntactic annotation). Indeed, it is an intrinsic property of natural languages to display legitimate variations. This is due to the fact that there are homonymous words in all natural languages. On the other hand, there are illegitimate cases, as exemplified in the previous section, since they originate from errors in annotation.

Thus, it would be interesting if we could find such erroneous variations, but this is a two-step problem: first we must be able to find the variations so we can – in a second step – proceed to distinguish the legitimate cases, i.e., *ambiguities*, from the erroneous ones. In this study, I am concerned with the first step, that of finding variation, letting the second step aside.

Dickinson & Meurers ([1]) extend the approach in [10] to syntactic annotation. In [10], they introduce the term *variation n-gram* for sequences of n tokens in a corpus that contain one or more tokens annotated differently in another occurrence of the same n-gram in the corpus. A token directly involved in the variation is called *variation nucleus*. Making use of the same concepts, the authors adapt the method in such a way that a variation nucleus is now the string (eventually unary) yielded by a constituent instead of tokens in isolation. The basic idea is to search for multiple occurrences of strings which were at least once analyzed as a constituent in the corpus.

Thus, for each instance of the string, its position in the corpus and syntactic label is recorded. However, some occurrences of a string may not form a constituent of its own. In Figure 4 bellow, I reproduce in a slightly simplified version an example from [1] where the second instance of “last month” does not form a constituent. In cases like this, the authors opted to assign a special label `NIL`.

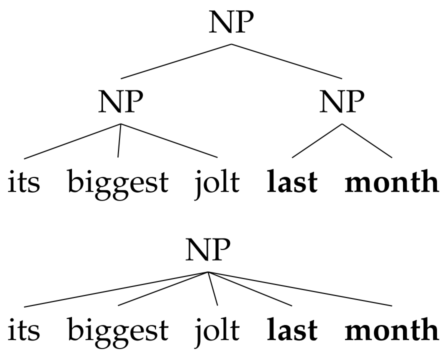

NP
NP
NP
its biggest jolt **last** **month**
NP
its biggest jolt **last** **month**

Figure 4. Constituent and “non-constituent” instances of a string (cf. [1]).

Some additional preprocessing is made to the corpus, before calculating the variation nuclei. First, functional annotation of syntactic categories is removed, since the authors believe that the method is better suited for categories than for syntactic functions. Thus, labels such as `NP-SBJ` (clause subject) or `IP-MAT` (root clause) used in the TBC annotation get stripped from their `-SBJ` and `-MAT` portions, respectively.

Sentences in the TBC and related treebanks include “null elements”, i.e., elements inserted in the trees in order to signal dislocation or omission of arguments or adjuncts, unstated units of measurement, and other things ([11]). For example, in the TBC one finds elements like `[NP-SBJ *pro*]` which signals the presence of a null subject. This kind of element is largely determined by theoretical assumptions and seems (cf. [1]) to be relatively independent of the local environment where it appears. Thus, the authors opted for not taking them into account in the calculation of variation nuclei.

One last modification is the collapsing of unary branching nodes, like `[NP [NPR John]]`, producing complex labels like `NP/NPR`. This modification was necessary in order to avoid redundancy since nodes related via unary branching yield the same string and will thus be considered variations of the same nucleus. Here, I decided to rule out such redundancies in a different way: the algorithm records the smaller tree that still yield the tokens of the variation nuclei.

After this preprocessing of the corpus, nuclei can finally be calculated. The authors propose the following algorithm to calculate the set of nuclei of length i ($1 \leq i \leq$ *length-of-longest-constituent-in-corpus*):

1. Compute the set of nuclei:
   a) Find all constituents of length i, store them with their category label.
   b) For each distinct type of string stored as a constituent of length i, add the label `NIL` for each non-constituent occurrence of that string.
2. Compute the set of variation nuclei by determining which of the nuclei were stored in step 1 with more than one label.

In addition to the steps above, the method generates the variation n-grams in the same way as that defined in [10]. The authors evaluate their method on the Wall Street Journal corpus (WSJ) – which is a million token part of the Penn Treebank-3 release ([12]) – and report that there were 34,564 variation nuclei found with sizes varying from 1 to 46. After applying heuristics for classification[2], a set of 6277 distinct variation nuclei was obtained.

From this output, 100 examples were randomly sampled and 71 were confirmed to be true errors. Taking the 95% confidence interval for the point estimate .71 (.6211, .7989) and applying it to the whole set of 6277 variation

2. The problem of classification is not part of the study presented here. In the method developed by the authors, two heuristics are used for the classification of syntactic variation nuclei as either errors or ambiguities. First, only n-grams for $n \geq 8$ are taken into account in order to maximize the probability of being an error. Second, the algorithm “distrusts the fringe”, that is, variation nuclei located at the fringe of variation n-grams are less likely to be errors and are thus taken as probable ambiguities.

nuclei gives us a number of errors between 3898 and 5014. As the authors note, however, this number would probably be larger, because each variation nuclei predicts *at least* one occurrence of an erroneous instance.

3.2 Alternative implementation

For the purposes of what follows, let us dub the original method as 'dm2003'. Let us call the alternative proposed here as 'dm2003-alt'. In 'dm2003', a given nucleus is taken as a variation if the algorithm can find at least two instances of it labeled by distinct categories, including the artificial label `NIL`. This is so despite of the internal structure of the syntactic annotations involved.

Two facts seem to undermine this strategy: first, it is absolutely possible that two instances of a nucleus have the same constituent label but distinct internal syntactic structures. As we can see in Figure 5, although both trees are syntactic variants of the nucleus, they will be considered the same since they have the same root label.

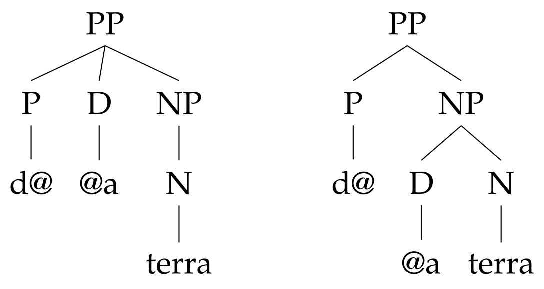


Figure 5. Internal syntactic inconsistency.

The second fact is that although not forming a constituent on its own, a particular instance of a nucleus may still be syntactically consistent with another instance. Take, for example, the trees in Figure 6. There are two instances of the nucleus "d@ @os homens", both consistent with each other on the structure assigned. The only difference is that the `NP` node in the second tree contains one more element which is irrelevant in this case: we are in the face of one and the same syntactic variant with respect to those three words. Nonetheless, 'dm2003' will consider them as distinct variants, with the second being labelled as `NIL`.

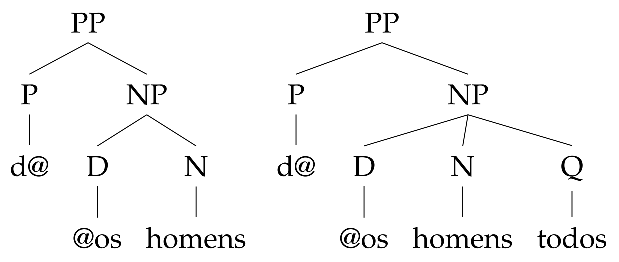


Figure 6. Consistency among two different trees.

Given these facts, in 'dm2003-alt' the strategy is to take the internal structure into account when comparing instances. Let us introduce the term "dominance chain" to make reference to the complete sequence of nodes from the root label to each terminal symbol in a tree. Thus, for the trees in Figure 5, we have the dominance chains in (1), while for the trees in Figure 6, we have (2):

(1) (i) `PP-P-d@`
`PP-D-@a`
`PP-NP-N-terra`
(ii) `PP-P-d@`
`PP-NP-D-@a`
`PP-NP-N-terra`

(2) (i) `PP-P-d@`
`PP-NP-D-@os`
`PP-NP-N-homens`
(ii) `PP-P-d@`
`PP-NP-D-@os`
`PP-NP-N-homens`
`PP-NP-Q-todos`

By comparing the dominance chains for the nuclei "d@ @a terra" and "d@ @os homens", we obtain the desired result of differentiating (1i) from (1ii) while equating (2i) and (2ii). The alternative algorithm for finding variation nuclei in 'dm2003-alt' is the following:

1. Compute the set of nuclei:
   a) Find all constituents of length i, *store them with their dominance chains*.
   b) For each distinct type of string stored as a constituent of length i, *find occurrences of that string with distinct sets of dominance*

chains.[3]

2. Compute the set of variation nuclei by determining which of the nuclei were stored in step 1 with *two or more distinct sets of dominance chains*.

4 Experimental results

In this section, results of the application of 'dm2003' and 'dm2003-alt' to the TBC are presented. The main goal of the experiments is to assess the relative precision of each method in finding *true* variations as well as in their relative recall, that is, checking whether the alternative method actually find more variations than the original. The results indicate that this is the case.

4.1 Corpus

The two methods were applied to the current publicized version of the TBC ([2]). It consists of 16 parsed texts which comprise a total[4] of 672,404 tokens distributed over 34,263 sentences. Two different runs of the algorithms were performed, one for the first 1000 sentences (16,951 tokens) from the parsed text file `a_001_psd.txt` and another run for the whole corpus.

4.2 Relative precision and recall

The total number of variation nuclei found by 'dm2003' for the partial corpus is 606 and the longest nuclei found has 5 tokens. The alternative 'dm2003-alt' found 202 variation nuclei, with maximum length of 4 tokens.[5] The two sets intersect in a total of 188 variation nuclei. Apart from these, all of the 418 cases exclusively captured as variation nuclei by the original method turned out to be confirmed as non-variants, in the strict sense adopted here. In other words, for each variation nucleus in those 418 cases, the "variants" have the same set of dominance chains and, thus, the same syntactic annotation.

This result demonstrates that the relative[6] precision of the original method for the partial corpus is only 31,02%. On the other hand, 14 variations of the 202 found by 'dm2003-alt' are true variants missed by the original method. This result demonstrate that the original method is not capable of finding all variation, showing a relative recall of 93,06% for the partial corpus.

For the whole corpus, results are similar. The method 'dm2003' found 26,363 variation nuclei with length up to 25 while 'dm2003-alt' found 10,780 variation nuclei with length up to 23. Figure 7 shows a 'four variant' variation nucleus found by the alternative method. The two sets intersect in 9,810 variation nuclei. Assuming that the remaining variations found by the original method are not true variations, as the results with the partial corpus indicate, we obtain a recall of 91%, similar but slightly lower than that for the partial corpus. The relative precision of the original method, on the other hand, increases for the whole corpus, reaching 37,21%.

4.3 Discussion

Both methods benefit from larger corpora since their sizes increase the chance for repetitions of token sequences to happen. In fact, if an error in annotation occurs for a non-repeating sequence of tokens it will never be found by the methods discussed here. But the above results seem to indicate that the original method is particularly dependent on the number and variety of inconsistencies in the corpus, that is, it depends on the fact that at least one variant of a given nucleus appears with a different label or as a non-constituent. The alternative method overcomes this limitation.

We must conclude that the low performance of the original method results from a misconception of what a syntactic variation is. As

3. The lengths of the dominance chains of a given occurrence are bounded by the first dominating node that yield it (exclusively or not).

4. Excluding empty elements and non-textual tokens tagged as CODE or ID.

5. Detailed outputs are available at: http://pablofaria.com.br/experiments/dhandes-output/.

6. Absolute measures are not possible here, since we lack the precise number of variations existent in the corpus. Thus, the measures for the original method are relative to the performance of the alternative implementation.

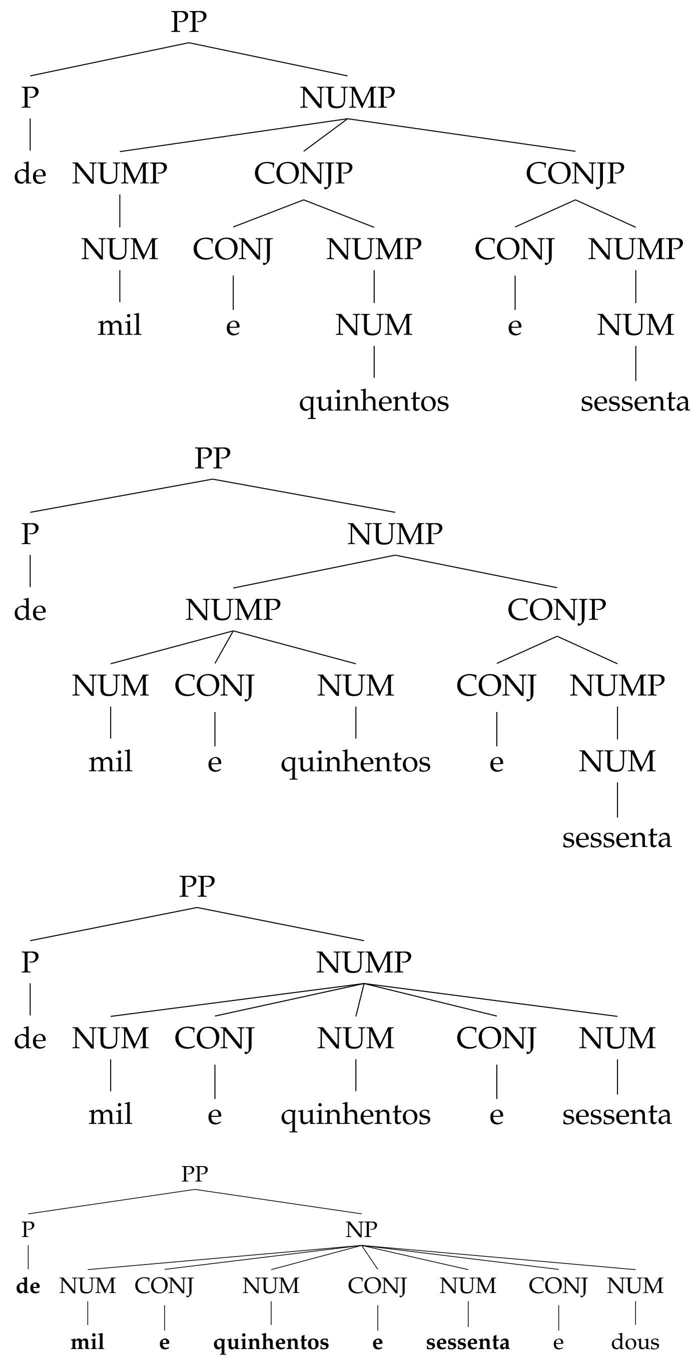


Figure 7. Variants of the nucleus “de mil e quinhentos e sessenta” found by the alternative method.

described in Section 3.1, a given nucleus is not considered as a variation if all of its instances have the same root label. Thus, differences in their internal structure are not taken into account. Dickinson & Meurers ([1]) suggest that the internal structure gets inspected when smaller nuclei are verified, but this does not solve the problem, since some of the variation in higher contexts in the tree continue to be missed. And this is particularly problematic for their method, because only variation nuclei with length $i \geq 8$ are (possibly) classified as errors.

4.4 The overall sensibility of the methods

The methods evaluated here are limited in many respects. First, they deliberately ignore functional sub-labels. Thus, errors such as the one shown on Figure 1 are not captured by these methods. An account of sub-labels, however, is not a matter of just inhibiting the step in the preprocessing of the corpus where labels get stripped of them. Since this kind of variation presuppose identical categorial contexts, they seem to demand an approach similar to the one sketched bellow for empty categories.

Although strongly determined by theory internal considerations, empty categories are still a possible locus of inconsistency and as such should also be analyzed. The intuition is that a variation relative to these elements must be characterized by the occurrence of two or more identical syntactic structures (regarding the non-empty elements) varying in the empty elements assigned.

Thus, it would take two steps in order to identify them, that is, one for the detection of identical non-empty instances of a nucleus and a further step to check whether these instances vary in their empty elements. This second step is not obvious, however, since it would demand a more substantial set of particular theoretical assumptions in order to determine the correct evaluation of these elements.

One last important source of limitation is the use of tokens as the units that constitute a nucleus of variation. It is well known that the lexicon is the main source of idiosyncrasies among languages. Although the performance seems to be satisfactory for languages like English or Brazilian Portuguese, for morphological richer languages the performance will decrease, maybe substantially, because repetitions of long sequences of tokens will be far more infrequent. These repetitions would also be infrequent for small corpora, despite the language involved.

The obvious move, in this case, would be to use part-of-speech tags instead of tokens as the relevant units. This is currently under investigation and the first results of runs to first 1000 sentences of the TBC indicate that a great deal of variation is possibly being obscured by lexical idiosyncrasies, since the number of variation nuclei found by 'dm2003-alt' is 413 which is more than two times greater than the number found using tokens.

Nonetheless, initial inspection of the variations found shows that more intricate questions arise when using tags and not all of the variation detected are in fact variation, specially when structures of adjunction and/or coordination are involved. A more robust processing of syntactic structures is needed in this case, such as those proposed in [4] and [5], for example, which apply explicit grammatical formalisms in order to describe tree structures.

5 Summary

As the number of initiatives concerned with treebank building grows up, detecting variations in annotated corpora becomes an increasingly important area of research. In this study, an implementation of the variation detection algorithm proposed in [1] was evaluated against the Tycho Brahe Corpus. Along with the original method, an alternative implementation differing in the way variation is defined was also evaluated.

It was demonstrated that the use of dominance chains instead of the root label for the characterization of variants improve the precision and recall of the procedure. This modification must be interpreted as a step towards a more adequate treatment of the data, given its particularities. Since we are dealing with syntactically annotated corpora, we should make use of all the information that such annotation can provide, for instance, dominance chains.

The decisive step, however, is to apply explicit grammatical formalisms in order to better describe the data and evaluate candidate variants. The complexity of syntactic structures need be taken into account since trees with quite distinct terminal nodes may still be equivalent to each other in higher domains of their (sub)structures. Thus, although terminal nodes are certainly a starting point for finding variants, they are not sufficient.

References

[1] M. Dickinson and W. D. Meurers, "Detecting inconsistencies in treebanks," in *Proceedings of TLT*, vol. 3, 2003, pp. 45–56.

[2] C. Galves and P. Faria. (2010) Tycho brahe parsed corpus of historical portuguese. [Online]. Available: http://goo.gl/cu4N6w

[3] Y. Matsumoto and T. Yamashita, "Using machine learning methods to improve quality of tagged corpora and learning models." in *LREC*, 2000.

[4] Y. Kato and S. Matsubara, "Correcting errors in a treebank based on synchronous tree substitution grammar," in *Proceedings of the ACL 2010 Conference Short Papers*, ser. ACLShort '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 74–79.

[5] S. Kulick, A. Bies, and J. Mott, "Using derivation trees for treebank error detection." in *ACL (Short Papers)*, 2011, pp. 693–698.

[6] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: the penn treebank," *Comput. Linguist.*, vol. 19, no. 2, pp. 313–330, Jun. 1993.

[7] C. Galves and H. Britto, "A construção do corpus anotado do português histórico tycho brahe: o sistema de anotação morfológica," in *Actas do IV Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR'99)*, I. Rodrigues and P. Quaresma, Eds., Évora, September 1999, pp. 81–90.

[8] T. McEnery and A. Hardie, *Corpus linguistics: method, theory and practice*. New York, NY, USA: Cambridge University Press, 2012.

[9] D. Blaheta, "Handling noisy training and testing data," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, ser. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 111–116.

[10] M. Dickinson and W. D. Meurers, "Detecting errors in part-of-speech annotation," in *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-03)*, Budapest, Hungary, 2003, pp. 107–114.

[11] C. Galves. (2008) Syntactic annotation system (of the tycho brahe corpus). [Online]. Available: http://goo.gl/YaR7g7

[12] M. Marcus, B. Santorini, M. A. Marcinkiewicz, and A. Taylor. (1999) Treebank-3 ldc99t42. Web Download. Philadelphia: Linguistic Data Consortium. [Online]. Available: https://catalog.ldc.upenn.edu/LDC99T42