What can we learn from natural and artificial dependency trees

Chunxiao Yan
Modyco
Université Paris Nanterre
CNRS - France
yanchunxiao@yahoo.fr

Marine Courtin
LPP
Université Paris 3 Sorbonne Nouvelle
CNRS - France
marine.courtin@sorbonne-nouvelle.fr

Abstract

This paper is centered around two main contributions : the first one consists in introducing several procedures for generating random dependency trees with constraints; we later use these artificial trees to compare their properties with the properties of natural trees (i.e trees extracted from treebanks) and analyze the relationships between these properties in natural and artificial settings in order to find out which relationships are formally constrained and which are linguistically motivated. We take into consideration five metrics: tree length, height, maximum arity, mean dependency distance and mean flux weight, and also look into the distribution of local configurations of nodes. This analysis is based on UD treebanks (version 2.3, Nivre et al. 2018) for four languages: Chinese, English, French and Japanese.

1 Introduction

We are interested in looking at the linguistic constraints on syntactic dependency trees to understand what makes certain structures plausible while others are not so plausible. To effectively do this kind of work, we need to observe natural trees (syntactic trees that are the results of linguistic analysis) to see what this population looks like. Similar work has been done for example by Jiang and Liu (2015) on the relation between sentence length, dependency distance and dependency direction. But observing natural trees only has its limits : we cannot see what is special about them and their properties, and we cannot distinguish the effects of the various constraints that affect them. We can only observe the structures that are the result of all these constraints and their interactions. On the other hand, if we start from a blank canvas, randomly generated trees, and incrementally add constraints on these trees, we might be able to study one by one the effects of each constraint, and to progressively add them to get closer to natural trees. Using artificially generated trees can also be insightful to determine which constraints are formally motivated (they are a result of the mathematical structure of the tree) and which constraints are linguistically or cognitively motivated. Research in the line of Gildea and Temperley (2009) who have used random and optimal linearisations to study dependency length and its varying degrees of minimization can help us to discover constraints that would be helpful to explain why we only find a small subset of all potential trees in syntactic analyses on real data.

Our objective is therefore twofold: first we want to see how different properties of syntactic dependency trees correlate, in particular properties that are related to syntactic complexity such as height, mean dependency distance and mean flux weight, then we want to find out if these properties can allow us to distinguish between artificial dependency trees (trees that have been manipulated using random components and constraints), and dependency trees from real data.

2 Looking into the properties of syntactic dependency trees

2.1 Features

In this work we use the five following metrics to analyze the properties of dependency trees:

Feature name	Description
Length	Number of nodes in the tree
Height	Number of edges between the root and its deepest leaf
Maximum arity	Maximum number of dependents of a node
Mean Dependency Distance (MDD)	Two nodes in a dependency relation are at distance 1 if they are neighbours, distance 2 if there is a node between them etc. For every tree, we look at the mean of those dependency distances. See (Liu 2008) for more information about this property.
Mean flux weight	Mean of the number of concomitant disjoint dependencies (Kahane et al. 2017; see comments below)

Table 1: Tree-based metrics

We chose these properties because we believe that they all interfere in linearization strategies, that is how words are ordered in sentences, and the effects of those linearisation strategies. Recently, there have been many quantitative works (Futrell et al., 2015; Liu 2008) that have focussed on dependency length and its minimization across many natural languages. In complement to these linear properties we also use "flux weight", a metrics proposed by Kahane et al. (2017) which captures the level of nestedness of a syntactic construction (the more nested the construction is, the higher its weight in terms of dependency flux). In their paper, they claim the existence of a universal upper bound for flux weight, as they have found it to be to 5 for 70 treebanks in 50 languages.

In addition to these tree-based metrics, we propose to look at local configurations using the linearised dependency trees. To look at these configurations, we extract and compare the proportion of all potential configurations of bigrams (two successive nodes) and trigrams (three successive nodes). For bigrams, we have three possible configurations: a -> b which indicates that a and b are linked with a relation on the right, a <- b which indicates that a and b are linked with a relation on the left, and a ♢ b, which indicates that a and b are not linked by a dependency. For trigram configurations (a, b, c), the possibility is much wider and we obtain 25 possible configurations. There are projective configurations like: a -> b -> c, (a -> b) & (a -> c), (a -> c) and (b <- c), but also non-projective cases like: a <- c and b -> c.

2.2 Hypotheses

In this section, we describe some of our hypotheses concerning the relationship between our selected properties. First, we expect to find that tree length is positively correlated with other properties. As the number of nodes increases, the number of possible trees increases including more complex trees with longer dependencies (which would increase MDD) and more nestedness (which would result in a higher mean flux weight). The relationship with maximum arity is less clear, as there could be an upper limit, which would make the relation between both of these properties non-linear. We are also particularly interested in the relationship between mean dependency distance and mean flux weight. An increase in nestedness is likely to result in more descendents being placed between a governor and its direct dependents, which would mean an overall increase in mean dependency distance.

For local configurations, we know that in natural trees, most of the dependencies occur between neighbours, see for example Liu (2008), the proportion varying depending on the language. It will be interesting to see how much that is still the case in the different random treebanks, depending on the added constraints.

For trigrams of nodes we are interested in the distribution of four groups of configurations that represent four different linearization strategies: "chain" subtrees that introduce more height in the dependency tree in with both dependents in the same direction, "balanced" subtrees that alternate dependents on both sides of the governor, "zigzag" subtrees which are similar to chains but with the second dependent going in the opposite direction as the first one, and "bouquet" subtrees where the two dependents are linked to the same governor (see examples in Figure A1 in the Appendix). If one group of configurations is preferred in natural trees compared to artificial ones, it could indicate that there exists some linguistic and/or cognitive constraints that make the configuration more likely to appear. We are also interested in the hypothesis advanced by Temperley (2008) who proposes that languages that strongly favor head-initial or head-final de-

pendencies will still tend to have some short phrases depending on the opposite direction, which could constitute a way of limiting dependency distances.

3 Random tree generation with constraints

In this section we will look at random dependency tree generation with constraints. We distinguish two different steps in the dependency tree generation process : the generation of the unordered structure, and the generation of the linearisation of the nodes. Throughout this generation process, we limited ourselves to projective trees. In order to compare the properties of natural and random trees we used 3 different tree generating algorithm, to which we assign the following names : original random (1), original optimal (2) and random random (3).

The first algorithm "original random" samples an unordered dependency structure from a treebank (i.e the original structure), and generates a random projective linearisation for it:

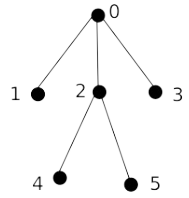


Figure 1: unordered tree

1. We start the linearisation at the root.
2. Then, we select its dependent nodes [1,2,3] and randomly order them, which gives us [2,1,3].
3. We select their direction at random, which gives us ["left", "left", "right"], and the linearisation steps [0], [20], [120], [1203].
4. We repeat steps 1 through 2 until every node has been linearized, which gives us (for example) [124503].

The second algorithm "original optimal" also samples an unordered dependency structure from a treebank, but instead of generating a simple projective linearisation, we add a second constraint to minimize dependency distances inside the linearised dependency tree. The idea comes from Temperley (2008): to minimize dependency distances in a projective setting, dependents of a governor should be linearized alternatively on opposing sides of the governor, with the smallest dependent nodes (i.e those that have the smallest number of direct and indirect descendants) linearized first. Using the same structure unordered tree as in fig. 1 we described the procedure below:

1. We start the linearisation at the root.
2. Then, we select its dependent node [1,2,3] and order them in order of their decreasing number of descendant nodes, which gives us [1,3,2].
3. We select a first direction at random, for example "left", and order these nodes alternating between left and right, which gives us these linearisation steps [0], [10], [103], [2103].
4. We repeat steps 1 through 2 until every node has been linearized, which gives us for example [425103].

The third algorithm "random random" is the only one to implement two random steps : first generate a completely random structure, then linearize it following the same procedure as in algorithm 1). The unordered structure generation step is described in fig 2.

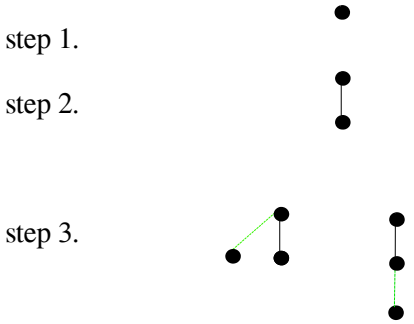


Figure 2: Random tree generation

1. We start the generation process with a single node
2. We introduce a new node and randomly draw its governor. For now, since there is only one potential governor, the edge has a probability of 1.
3. We introduce a new node and randomly draw its governor. There are two potential governors which gives us a probability 0.5 of drawing the node 0 and the same probability for the node 1. These potential edges are drawn in green on the graph.
4. We repeat this last step until all nodes have been drawn and attached to their governor.[1].

These tree generation algorithms are only some of the many possible algorithms that could be implemented, but they us give us tools to analyze how different generation strategies will affect the properties of the generated trees, as we incorporate more and more constraints into the two generation steps. They are also easily extensible, for example during the linearisation process we could introduce a probability of creating a head-final edge, to produce trees that resemble more the trees of a head-final language like Japanese. For the unordered structure generation, we could introduce a constraint to limit length, arity or height. We need to distinguish constraints that happen during the unordered structure generation step and constraints that have to do with linearisation, like constraints on dependency distances and on flux weights.

One question that still remains concerns the ordering of the two steps : unordered structure generation and linearisation generation. So far we have only implemented the full generation starting with the generation of the unordered structure and then moving on to the linearisation, which is a synthesis approach as described in Meaning-Text-Theory (Mel'čuk 1998), but it would be interesting to go in the analysis direction, starting with a sequence of nodes, and then randomly producing a structure for it. This could allow us to see how generation algorithms impact the distribution of trees, especially as we add constraints into the generation. We could then see if one type of random generation (synthesis vs analysis) produces structures that resemble natural dependency structures more, or if they introduce biases towards some types of structures.

4 Results and discussion

4.1 Correlation between properties

For each pair of properties presented in section 2.1 we measured the pearson correlation coefficient to find out the extent to which the relationship between these variables can be linearly captured. We looked into these results for the different natural treebanks (“original”) and the artificial ones (“original random”, “random random” and “original optimal”). Tables presenting the full results are showcased in tables 1-4 in appendix, with rankings for the correlation between parentheses.

Based on these results, we notice that mean dependency distance and mean flux weight are overall the most correlated properties with values ranging from 0.70 (jp_pud, “original”) to 0.95 (fr_partut, “original optimal”). This can be explained by the fact that mean flux weight increases as the number of disjoint flux increases, which in turn tends to create longer dependencies than structure with few disjoint flux. An interesting observation about this correlation is that it is intensified in all the artificial treebanks, and is the strongest in the “original optimal” version. Introducing a dependency distance minimization constraint will

[1] Note that this algorithm gives us a uniform probability on derivations, but that some derived trees are more probable than others, for example if the length of the tree is 4 we only have 1 derivation to obtain a tree of height 4, and 2 derivations to obtain a tree with 2 dependent on the root and 1 on one of these dependents.

favour shorter dependencies, which provides less opportunities for configurations that introduce disjoint flux. Therefore the mean flux weight will also decrease.

If we look at the the correlation between length and height, we find that it is strong in original structures (0.78 correlation) as well as in the random ones (0.71 correlation in "random random", which is the only format in which the height of the tree is affected by the manipulation). This means that the relationship between these two properties is not motivated by linguistic factors only. From a mathematical point of view, longer sentences have the potential to introduce more hierarchy which increases the height. Thus, there is a correlation between these two properties regardless of wether the structure is natural or random. Zhang and Liu (2018) have proposed that the relationship between these two properties in natural treebanks of English and Chinese can be described by a power law. Further examination could tell us if it is also the case for randomly generated trees, or if the relationship is better modelled by another type of function.

We also find quite strong correlations between mean dependency distance and height in the artificial treebanks (0.76, 0.79, 0.72 respectively for "original random", "original optimal" and "random random") while this correlation is less important for the natural treebanks (0.46). It is quite interesting that the correlation decreases in the original trees. Our interpretation is that perhaps there is a more complex relationship at play between height and mean dependency distance in real data that cannot be linearly captured, and this complex relationship would be altered by the random components when generating the various artificial trees, especially as we relinearize the nodes.

4.2 Distribution of configurations

In this section we look at the distribution of local syntactic configurations by extracting trigrams and looking at their dependency relations. First we look at the non-linearized configurations : a→b→c and b←a→c, to analyze the differences in local structures between natural and randomly generated trees. Then we analyze the distribution of the four different groups presented in section 2.2, and how tis distribution is impacted by language and the type of treebank (natural and artificial). We will discuss here a few points and present the full results in appendix.

In fig 3, we can see the distribution of non-linearized configurations for one example language, French. For the "random random" trees, we have 45% of b←a→c configurations and 55% of a→b→c configurations. For all other treebank types, the first configuration is by far the most frequent one. This will likely have some repercussions in the distribution of linearized configurations.

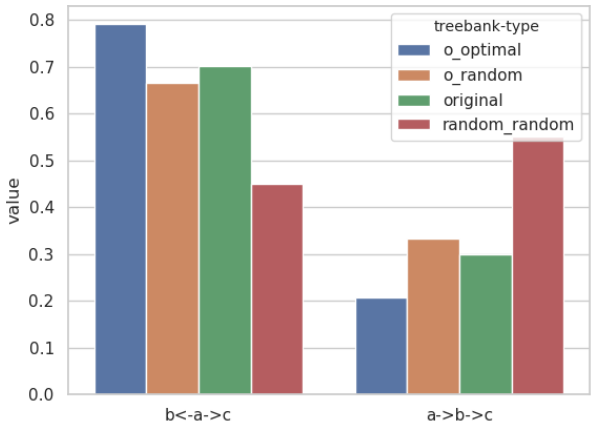


Figure 3: Non-linearized trigram configurations distribution for French

We also observe that the results are fairly similar across all 4 languages, with "original optimal" showing the most unequal distribution (80%-20% respectively for b←a→c and a→b→c configurations), followed by "original" and "original random" (around 60%-40%, although there is some variation depending on the language). One possible explanation for favouring b←a→c could be that it helps minimizing dependency distances, since it can lead to "balanced" configurations which are the optimal way to arrange dependents whithout introducing longer dependencies. If that is the case, we will see a high proportion of "balanced" configurations when we look more in detail at how these configurations are linearized. Another line of explanation could be that having too many a→b→c configurations introduces too much height in the trees, which could be a factor of complexity that natural languages try to avoid whenever possible. Differences

between “original optimal”, “original random” and “original” can be explained by the linearization process: the optimal trees tend to favour shorter dependencies, which means that a higher percentage of triplets of nodes will all be connex, while non-optimal trees will sometimes linearize the nodes further away, thus excluding them from the extraction of triplets. It would be interesting to see if the distribution is similar when we look at all configurations of triplets and not just at local ones.

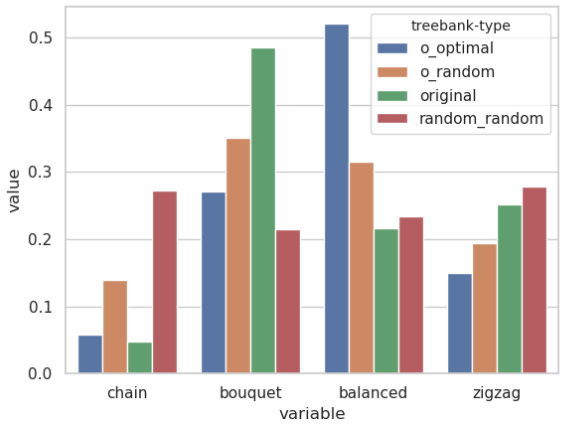


Figure 4: Trigram configurations distribution for French

We then go on to look at these configurations once they have been subdivided according to the classification proposed in section 2.2. Note that the configurations “bouquet” and “balanced” are a result of the b←a→c configurations and that a→b→c will produce either “chain” or “zigzag”. We show the distribution for French in fig 4. First we comment the results that are stable across languages: “random random” trees have a slight preference for “chain” and “zigzag” as a result of the preference for b←a→c configurations, but inside each group (“chain” and “zigzag” / “bouquet” and “balanced”) the distribution is equally divided. The “original optimal” trees have a very marked preference for “balanced” which is to be expected because alternatively ordering dependents of a governor is the prefered strategy to minimize dependency length. Next we find “zigzag” configurations, followed by “bouquet” and very few “chain”. Contrary to the potential explanation we advanced for the high frequency of b←a→c configurations, “balanced” configurations are not particularly frequent in the original trees (23% in Chinese, 14% in English, 21% in French and 27% in Japanese), especially when compared to the “bouquet” configurations (37%, 52%, 48%, 30% respectively). Bouquet configurations are much more frequent in the natural trees than in the artificial ones. We have yet to find a satisfactory explanation for this. Even if we know that some arbitrary choices in the UD annotation scheme inflate the percentage of bouquet (*conj*, *fixed* and *flat* relations are always encoded as a bouquet), this does not seem sufficient to explain the difference with the other configurations. We also remark that, if we were to use a schema with functional heads most of these “bouquet” configurations would become “zigzagz” or “chain”, so we could potentially find an explanation by investigating there. For the optimal model, the bouquet is not an optimal strategy to minimize dependency distances, so the bouquet configuration will, of course, be less critical in the optimal model.

Compared to the other languages, Japanese has an interestingly high percentage of “zigzag” configurations. This can be partly explained by the segmentation used in the Japanese treebanks. The particles and agglutinated markers (for polarity, aspect, politenesss...) have been annotated as separate tokens, which often creates many dependents on a single governor. A lot of these dependencies fall outside the trigram windows and are excluded from our analysis. Japanese being a head-final language, the configurations captured will often contain a head-final dependency (*obj*, *acl*, *nmod*...) and a marker of the dependent, which means that it will often fall into the “zigzag” bin. Nonetheless “bouquet” are still quite frequent as a governor often has several marks, and “balanced” capture nominal modifiers or compounds, and their case or topic marker.

5 Conclusion

In this paper we introduced several ways to generate artificial syntactic dependency trees and proposed to use those trees as a way of looking into the structural and linguistic constraints on syntactic structures for 4 different languages. We propose to incrementally add constraints on these artificial trees to observe the ef-

fects these constraints produce and how they interact with each other. We limited ourselves to generating projective trees, which we now realize was a very strong constraint that strongly restricts the types of structures available, and therefore the variations of the different observed properties, and think that it would be interesting to also look at the result when allowing non-projective edges.

To expand on this work we would also like to see how the observed properties and the relations between them are affected by the annotation scheme, in particular contrasting schemas where content words are governors (as is the case in UD) and schemas where function words are governors (for example using the SUD schema proposed by Gerdes et al. (2018)), as it will have an impact on height, dependency distances, and the types of configurations that can be extracted from the treebanks.

In the present paper, we have looked at local syntactic configurations through the extraction of sequences of nodes (pairs and triplets). However these configurations are not representative of all configurations inside the trees, as some syntactic relations are more likely to appear in more global configurations. In the future, we plan on looking at these larger configurations by extracting subtrees and analyzing their distribution. We also intend on digging deeper into the analysis of the present data, and propose predictive models that could help us clarify the relationship (whether they be linear or not) between the different features in order to build a more solid basis to verify our hypotheses and propose explanations for the observations we made.

References

Richard Futrell, Kyle Mahowald and Edward Gibson. 2015. Large-scale evidence of dependency length minimization in 37 languages. *Proceedings of the National Academy of Sciences*, 112(33), 10336–10341. https://doi.org/10.1073/pnas.1502134112

Daniel Gildea and David Temperley. 2010. Do Grammars Minimize Dependency Length? *Cognitive Science*, 34(2), 286–310. https://doi.org/10.1111/j.1551-6709.2009.01073.x

Haitao Liu. 2008. Dependency Distance as a Metric of Language Comprehension Difficulty. *Journal of Cognitive Science, 9*(2), 159–191. https://doi.org/10.17791/jcs.2008.9.2.159

Jingyang Jiang and Haitao Liu. 2015. The effects of sentence length on dependency distance, dependency direction and the implications–based on a parallel English–Chinese dependency treebank. *Language Sciences 50* (2015: 93-104).

Sylvain Kahane, Chunxiao Yan and Marie-Amélie Botalla. 2017. What are the limitations on the flux of syntactic dependencies? Evidence from UD treebanks. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling2017)* (pp.73-82).

Igor Mel'čuk. 1998. *Dependency syntax: Theory and Practice*. SUNY press.

Joakim Nivre, Mitchell Abrams, Željko Agić et al. 2018. Universal Dependencies 2.3, LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, http://hdl.handle.net/11234/1-2895.

David Temperley. 2008. Dependency-length minimization in natural and artificial languages∗. *Journal of Quantitative Linguistics*, *15*(3), 256–282. https://doi.org/10.1080/09296170802159512

Hongxin Zhang and Haitao Liu. 2018. Interrelations among Dependency Tree Widths, Heights and Sentence Lengths. In: Jingyang Jiang & Haitao Liu (eds.). *Quantitative Analysis of Dependency Structures*, Berlin/Boston: DE GRUYTER MOUTON. pp. 31-52.

Appendix

original treebanks

	height_length	arity_length	arity_height	mdd_length	mdd_height	mdd_arity	mfw_length	mfw_height	mfw_arity	mfw_mdd
en_lines	0.77 (1)	0.62 (6)	0.37 (10)	0.68 (4)	0.44 (8)	0.72 (3)	0.65 (5)	0.49 (7)	0.42 (9)	0.77 (2)
en_gum	0.8 (2)	0.68 (4)	0.5 (10)	0.68 (5)	0.53 (9)	0.79 (3)	0.64 (6)	0.57 (7)	0.56 (8)	0.83 (1)
en_esl	0.71 (2)	0.57 (6)	0.2 (10)	0.62 (5)	0.23 (9)	0.65 (3)	0.62 (4)	0.33 (7)	0.32 (8)	0.72 (1)
en_partut	0.76 (2)	0.55 (6)	0.3 (10)	0.57 (5)	0.31 (9)	0.6 (3)	0.58 (4)	0.37 (7)	0.33 (8)	0.78 (1)
en_ewt	0.82 (3)	0.72 (4)	0.61 (10)	0.7 (5)	0.64 (9)	0.85 (2)	0.65 (8)	0.66 (7)	0.68 (6)	0.87 (1)
en_pud	0.65 (2)	0.44 (6)	0.09 (9)	0.49 (3)	0.08 (10)	0.48 (5)	0.48 (4)	0.19 (7)	0.13 (8)	0.72 (1)
fr_gsd	0.73 (2)	0.52 (6)	0.22 (10)	0.61 (5)	0.28 (9)	0.65 (3)	0.63 (4)	0.37 (7)	0.32 (8)	0.74 (1)
fr_sequoia	0.81 (3)	0.69 (4)	0.56 (9)	0.68 (5)	0.54 (10)	0.82 (2)	0.67 (6)	0.59 (8)	0.63 (7)	0.83 (1)
fr_spoken	0.84 (1)	0.59 (6)	0.42 (10)	0.61 (5)	0.44 (9)	0.78 (2)	0.67 (4)	0.56 (7)	0.46 (8)	0.71 (3)
fr_partut	0.79 (1)	0.54 (6)	0.35 (10)	0.61 (5)	0.37 (8)	0.67 (3)	0.65 (4)	0.47 (7)	0.37 (9)	0.77 (2)
fr_pud	0.64 (2)	0.47 (6)	0.1 (10)	0.58 (3)	0.17 (9)	0.54 (5)	0.56 (4)	0.29 (7)	0.21 (8)	0.77 (1)
zh_cfl	0.76 (2)	0.61 (5)	0.37 (10)	0.58 (7)	0.52 (8)	0.63 (4)	0.6 (6)	0.66 (3)	0.42 (9)	0.78 (1)
zh_gsd	0.58 (6)	0.56 (7)	0.14 (10)	0.61 (4)	0.39 (8)	0.65 (2)	0.63 (3)	0.6 (5)	0.27 (9)	0.74 (1)
zh_pud	0.57 (3)	0.53 (6)	0.07 (10)	0.56 (5)	0.41 (8)	0.52 (7)	0.56 (4)	0.59 (2)	0.19 (9)	0.77 (1)
zh_hk	0.83 (2)	0.73 (6)	0.56 (10)	0.79 (4)	0.72 (7)	0.79 (3)	0.69 (8)	0.74 (5)	0.61 (9)	0.86 (1)
jp_pud	0.58 (4)	0.47 (6)	0.0 (10)	0.59 (3)	0.13 (9)	0.59 (2)	0.54 (5)	0.36 (7)	0.17 (8)	0.7 (1)
jp_gsd	0.74 (2)	0.61 (6)	0.31 (10)	0.69 (4)	0.37 (9)	0.7 (3)	0.66 (5)	0.48 (7)	0.4 (8)	0.8 (1)
jp_modern	0.85 (1)	0.62 (4)	0.46 (9)	0.58 (6)	0.44 (10)	0.72 (3)	0.61 (5)	0.58 (7)	0.5 (8)	0.81 (2)

original_optimal

	height_length	arity_length	arity_height	mdd_length	mdd_height	mdd_arity	mfw_length	mfw_height	mfw_arity	mfw_mdd
jp_gsd	0.74 (4)	0.61 (7)	0.31 (9)	0.76 (3)	0.74 (5)	0.57 (8)	0.68 (6)	0.79 (2)	0.26 (10)	0.89 (1)
jp_pud	0.58 (5)	0.47 (7)	0.01 (10)	0.64 (3)	0.59 (4)	0.36 (8)	0.57 (6)	0.71 (2)	0.05 (9)	0.88 (1)
jp_modern	0.85 (4)	0.62 (7)	0.46 (9)	0.81 (5)	0.86 (3)	0.6 (8)	0.79 (6)	0.88 (2)	0.41 (10)	0.94 (1)
en_esl	0.71 (4)	0.57 (7)	0.2 (9)	0.73 (3)	0.7 (5)	0.48 (8)	0.63 (6)	0.75 (2)	0.16 (10)	0.89 (1)
en_ewt	0.82 (4)	0.72 (7)	0.61 (10)	0.8 (6)	0.84 (2)	0.8 (5)	0.69 (8)	0.82 (3)	0.63 (9)	0.93 (1)
en_partut	0.76 (4)	0.55 (7)	0.3 (9)	0.76 (5)	0.77 (3)	0.47 (8)	0.71 (6)	0.8 (2)	0.27 (10)	0.94 (1)
en_lines	0.77 (4)	0.62 (8)	0.37 (9)	0.79 (3)	0.77 (5)	0.63 (7)	0.72 (6)	0.8 (2)	0.35 (10)	0.9 (1)
en_gum	0.8 (4)	0.68 (8)	0.5 (9)	0.79 (5)	0.81 (3)	0.7 (7)	0.7 (6)	0.81 (2)	0.49 (10)	0.92 (1)
en_pud	0.65 (4)	0.44 (7)	0.08 (9)	0.68 (3)	0.62 (5)	0.35 (8)	0.61 (6)	0.69 (2)	0.07 (10)	0.89 (1)
zh_gsd	0.58 (5)	0.56 (6)	0.13 (10)	0.72 (2)	0.55 (7)	0.53 (8)	0.65 (4)	0.65 (3)	0.21 (9)	0.86 (1)
zh_cfl	0.76 (2)	0.61 (8)	0.38 (10)	0.75 (4)	0.69 (6)	0.68 (7)	0.7 (5)	0.75 (3)	0.39 (9)	0.87 (1)
zh_hk	0.83 (2)	0.73 (6)	0.57 (9)	0.81 (4)	0.79 (5)	0.81 (3)	0.62 (8)	0.73 (7)	0.55 (10)	0.87 (1)
zh_pud	0.58 (5)	0.53 (7)	0.08 (9)	0.65 (3)	0.6 (4)	0.39 (8)	0.54 (6)	0.68 (2)	0.07 (10)	0.86 (1)
fr_sequoia	0.81 (4)	0.69 (8)	0.56 (10)	0.8 (5)	0.83 (2)	0.72 (7)	0.73 (6)	0.83 (3)	0.56 (9)	0.94 (1)
fr_gsd	0.73 (4)	0.52 (7)	0.22 (10)	0.74 (3)	0.73 (5)	0.44 (8)	0.69 (6)	0.78 (2)	0.22 (9)	0.92 (1)
fr_partut	0.78 (4)	0.54 (7)	0.35 (9)	0.75 (5)	0.81 (3)	0.47 (8)	0.71 (6)	0.82 (2)	0.31 (10)	0.95 (1)
fr_spoken	0.84 (3)	0.59 (8)	0.42 (9)	0.82 (4)	0.81 (5)	0.67 (7)	0.77 (6)	0.84 (2)	0.32 (10)	0.88 (1)
fr_pud	0.64 (5)	0.47 (7)	0.1 (10)	0.68 (3)	0.65 (4)	0.36 (8)	0.63 (6)	0.72 (2)	0.14 (9)	0.92 (1)

original_random

	height_length	arity_length	arity_height	mdd_length	mdd_height	mdd_arity	mfw_length	mfw_height	mfw_arity	mfw_mdd
fr_pud	0.64 (3)	0.47 (7)	0.1 (10)	0.63 (4)	0.62 (5)	0.37 (8)	0.61 (6)	0.71 (2)	0.2 (9)	0.85 (1)
fr_spoken	0.84 (3)	0.59 (8)	0.42 (9)	0.82 (4)	0.8 (6)	0.6 (7)	0.81 (5)	0.86 (2)	0.4 (10)	0.9 (1)
fr_partut	0.79 (4)	0.54 (8)	0.36 (10)	0.78 (5)	0.79 (3)	0.55 (7)	0.76 (6)	0.85 (2)	0.41 (9)	0.92 (1)
fr_gsd	0.73 (3)	0.52 (7)	0.22 (10)	0.71 (4)	0.69 (6)	0.45 (8)	0.7 (5)	0.78 (2)	0.26 (9)	0.88 (1)
fr_sequoia	0.81 (4)	0.69 (8)	0.56 (10)	0.81 (5)	0.82 (3)	0.69 (7)	0.78 (6)	0.86 (2)	0.59 (9)	0.93 (1)
jp_gsd	0.74 (4)	0.61 (7)	0.31 (10)	0.75 (3)	0.71 (6)	0.57 (8)	0.73 (5)	0.8 (2)	0.37 (9)	0.87 (1)
jp_modern	0.85 (4)	0.62 (7)	0.46 (10)	0.85 (3)	0.83 (6)	0.6 (8)	0.84 (5)	0.87 (2)	0.47 (9)	0.93 (1)
jp_pud	0.58 (4)	0.47 (7)	0.0 (10)	0.6 (3)	0.54 (6)	0.39 (8)	0.57 (5)	0.71 (2)	0.07 (9)	0.78 (1)
en_esl	0.71 (3)	0.57 (7)	0.2 (10)	0.69 (4)	0.64 (6)	0.44 (8)	0.65 (5)	0.74 (2)	0.23 (9)	0.85 (1)
en_pud	0.65 (3)	0.44 (7)	0.08 (10)	0.6 (5)	0.62 (4)	0.33 (8)	0.6 (6)	0.71 (2)	0.13 (9)	0.85 (1)
en_partut	0.76 (3)	0.55 (7)	0.3 (10)	0.73 (6)	0.74 (4)	0.48 (8)	0.73 (5)	0.81 (2)	0.33 (9)	0.9 (1)
en_gum	0.8 (3)	0.68 (7)	0.5 (10)	0.79 (4)	0.77 (5)	0.67 (8)	0.76 (6)	0.82 (2)	0.55 (9)	0.9 (1)
en_ewt	0.82 (4)	0.72 (8)	0.61 (10)	0.81 (5)	0.82 (3)	0.76 (7)	0.77 (6)	0.86 (2)	0.67 (9)	0.92 (1)
en_lines	0.77 (4)	0.62 (7)	0.37 (10)	0.77 (3)	0.74 (6)	0.58 (8)	0.75 (5)	0.8 (2)	0.42 (9)	0.9 (1)
zh_gsd	0.58 (5)	0.56 (6)	0.13 (10)	0.66 (2)	0.48 (8)	0.52 (7)	0.64 (3)	0.63 (4)	0.23 (9)	0.8 (1)
zh_hk	0.83 (2)	0.73 (8)	0.56 (10)	0.82 (3)	0.79 (5)	0.75 (6)	0.74 (7)	0.8 (4)	0.6 (9)	0.89 (1)
zh_cfl	0.76 (3)	0.61 (8)	0.37 (10)	0.75 (4)	0.67 (6)	0.63 (7)	0.71 (5)	0.76 (2)	0.42 (9)	0.85 (1)
zh_pud	0.57 (5)	0.53 (6)	0.07 (10)	0.63 (2)	0.47 (7)	0.45 (8)	0.59 (4)	0.62 (3)	0.18 (9)	0.81 (1)

random_random

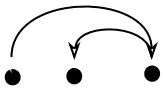
	height_length	arity_length	arity_height	mdd_length	mdd_height	mdd_arity	mfw_length	mfw_height	mfw_arity	mfw_mdd
zh_gsd	0.61 (5)	0.53 (7)	0.14 (10)	0.65 (3)	0.55 (6)	0.42 (8)	0.64 (4)	0.65 (2)	0.23 (9)	0.85 (1)
zh_hk	0.8 (3)	0.75 (7)	0.58 (10)	0.8 (4)	0.79 (5)	0.73 (8)	0.75 (6)	0.81 (2)	0.63 (9)	0.91 (1)
zh_pud	0.57 (5)	0.54 (6)	0.14 (10)	0.62 (3)	0.54 (7)	0.44 (8)	0.61 (4)	0.62 (2)	0.25 (9)	0.85 (1)
zh_cfl	0.72 (5)	0.6 (8)	0.37 (10)	0.77 (2)	0.68 (6)	0.64 (7)	0.75 (4)	0.76 (3)	0.47 (9)	0.88 (1)
fr_pud	0.57 (4)	0.51 (7)	0.11 (10)	0.59 (3)	0.52 (6)	0.41 (8)	0.57 (5)	0.61 (2)	0.19 (9)	0.83 (1)
fr_partut	0.68 (6)	0.65 (7)	0.42 (10)	0.72 (3)	0.68 (5)	0.61 (8)	0.7 (4)	0.75 (2)	0.48 (9)	0.89 (1)
fr_spoken	0.75 (5)	0.68 (7)	0.49 (10)	0.76 (3)	0.74 (6)	0.66 (8)	0.76 (4)	0.79 (2)	0.52 (9)	0.9 (1)
fr_sequoia	0.75 (5)	0.71 (8)	0.6 (10)	0.77 (4)	0.79 (3)	0.73 (7)	0.75 (6)	0.83 (2)	0.63 (9)	0.92 (1)
fr_gsd	0.63 (5)	0.58 (7)	0.23 (10)	0.68 (3)	0.59 (6)	0.5 (8)	0.66 (4)	0.68 (2)	0.31 (9)	0.86 (1)
en_lines	0.71 (5)	0.65 (7)	0.4 (10)	0.74 (3)	0.68 (6)	0.6 (8)	0.72 (4)	0.75 (2)	0.45 (9)	0.89 (1)
en_pud	0.58 (5)	0.52 (6)	0.08 (10)	0.61 (3)	0.51 (7)	0.4 (8)	0.6 (4)	0.64 (2)	0.18 (9)	0.82 (1)
en_partut	0.65 (5)	0.59 (7)	0.28 (10)	0.66 (3)	0.61 (6)	0.52 (8)	0.66 (4)	0.72 (2)	0.36 (9)	0.86 (1)
en_ewt	0.77 (6)	0.75 (7)	0.66 (10)	0.79 (4)	0.82 (3)	0.77 (5)	0.74 (8)	0.85 (2)	0.7 (9)	0.93 (1)
en_esl	0.63 (5)	0.57 (6)	0.17 (10)	0.67 (2)	0.57 (7)	0.47 (8)	0.65 (4)	0.66 (3)	0.27 (9)	0.85 (1)
en_gum	0.73 (6)	0.71 (7)	0.53 (10)	0.77 (3)	0.75 (4)	0.69 (8)	0.74 (5)	0.8 (2)	0.59 (9)	0.91 (1)
jp_gsd	0.69 (5)	0.64 (7)	0.37 (10)	0.73 (3)	0.66 (6)	0.59 (8)	0.72 (4)	0.74 (2)	0.44 (9)	0.89 (1)
jp_pud	0.52 (5)	0.5 (6)	0.06 (10)	0.56 (3)	0.48 (7)	0.36 (8)	0.53 (4)	0.6 (2)	0.13 (9)	0.82 (1)
jp_modern	0.7 (5)	0.69 (7)	0.46 (10)	0.72 (3)	0.69 (6)	0.67 (8)	0.7 (4)	0.78 (2)	0.54 (9)	0.9 (1)

Examples of 4 types of trigram configurations

1. Balanced

2. Chain

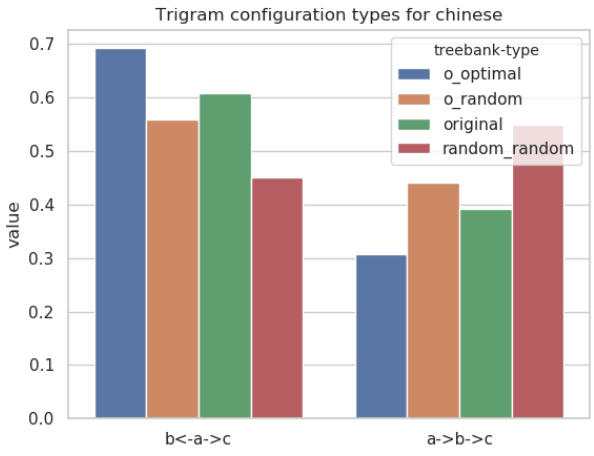
3. Zigzag

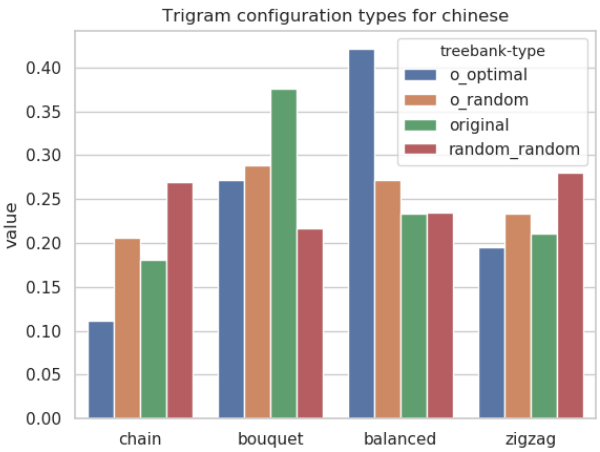


4. Bouquet

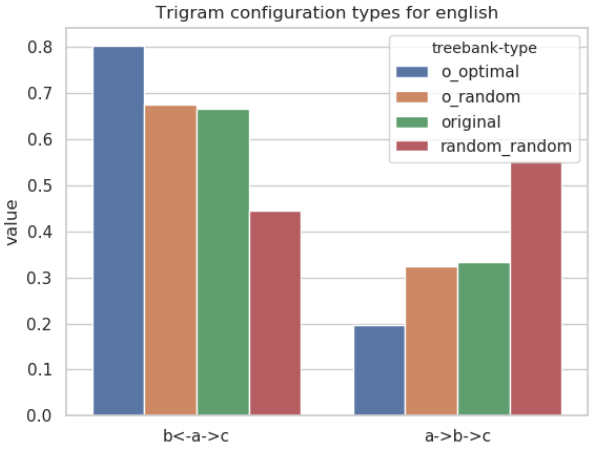
Trigrams configurations by type

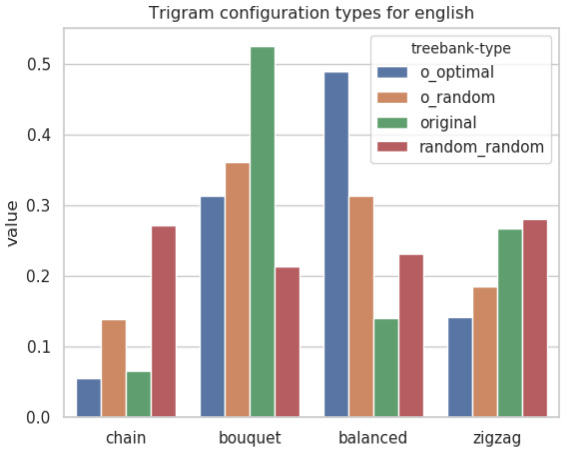
Chinese



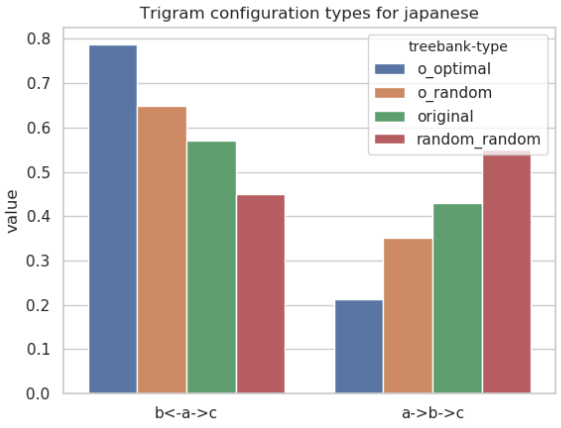


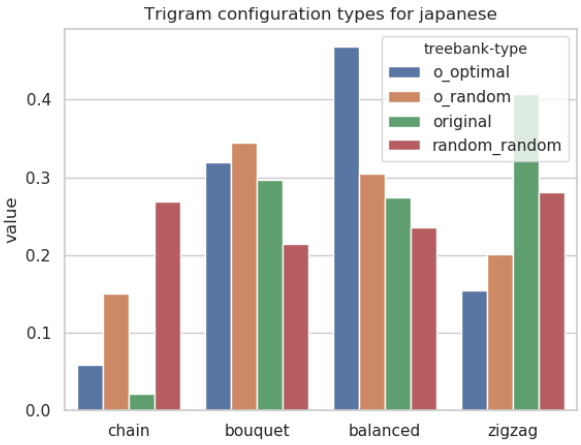
English


Trigram configuration types for english
treebank-type
o_optimal
o_random
original
random_random
value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
b<-a->c
a->b->c


Trigram configuration types for english
treebank-type
o_optimal
o_random
original
random_random
value
0.0
0.1
0.2
0.3
0.4
0.5
chain
bouquet
balanced
zigzag

Japanese


Trigram configuration types for japanese
treebank-type
o_optimal
o_random
original
random_random
value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
b<-a->c
a->b->c


Trigram configuration types for japanese
treebank-type
o_optimal
o_random
original
random_random
value
0.0
0.1
0.2
0.3
0.4
chain
bouquet
balanced
zigzag