What can we learn from natural and artificial dependency trees

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 treebanks to compare the properties of these random trees with natural trees (i.e trees extracted from treebanks) and analyze the relationships between these properties in both cases in order to better understand which relationships are formally constrained and which relationships 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 configurations for pairs and triples of nodes. This analysis is based on UD treebanks for four languages : English, Japanese, Chinese and French

Introduction

We're interested in understanding 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. But looking at natural trees can only tell us so much, we still don't get a thorough understanding of the relationship between their properties, and how these properties affect one another. On the other hand, if we start from a blank canvas, randomly generated trees and incrementally add constraints on these trees, we might closer and closer to natural trees, and we can use those trees to determine which constraints are formally motivated (they are a result of the mathematical structure of the tree) and which constraints are linguistically and cognitively motivated. These constraints help to explain why we only find a small subset of all potential trees in syntactic analyses.

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 natural languages.

Looking into the properties of syntactic dependency trees

In this work we use the five following metrics to analyze the properties of dependency trees :based on 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	The maximum number of dependents of a node inside the tree
Mean Dependency Distance (MDD)	Two nodes in a governor-dependent relation are at distance one if they are neighbours, distance two 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 flux weight introduced by Kahane et al. (2017)

We chose these properties because we believe that they all have something to do with linearization strategies, that is how words are ordered in sentences, and the effects of those linearization strategies. Recently, there has 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 metric 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 have shown that there could be an 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 are also interested in studying 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 in a governor dependent relation. For trigram configurations (a, b, c), the possibility is much wider, we have 27 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.

Some of our 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 possibles 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're 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 (see graph below)

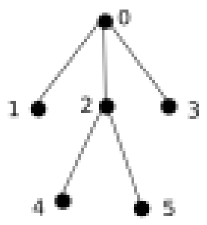
For local configurations, we know that in natural trees, most of the governor dependent relations occur between a node and its neighbours, the most frequent dependency distance being one, and the second most frequent two. It will be interesting to see how much that is still the case in the different random treebanks, depending on the added constraints.

For trigrams of nodes we are interested in particular in the distribution of four groups of configurations that represent four different strategies of linearization : "chained" subtrees that introduce more depth in the dependency tree in one direction, "balanced" subtrees that alternate dependents on both sides of the governor, "contresens" subtrees which are similar to chains but with the 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. We included an example of each of these groups of configurations in annex. If one group of configurations is preferred, it could indicate that this linearisation strategy is more economical. We're also interested in the hypothesis advanced by [Temperley 2008] who proposes that languages that strongly favor head-initial or head-final dependencies will still tend to have some short dependents on the opposite direction, which could constitute a way of limiting dependency distances.

Random tree generation with constraints

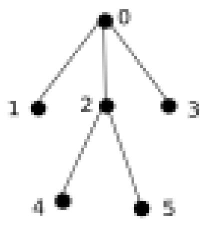
In this section we will look at random dependency tree generation with constraints. We distinguish two different steps in the dependency tree generation process : the generation of the unordered structure, and the generation of the linearisation of the nodes. In order to compare the properties of natural 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).

1) The first algorithm "original random" samples an unordered dependency structure from a treebank (i.e the original structure), and generates a random linearisation for it. We imposed a projectivity constraint on this linearisation and proposed the following algorithm to generate it:



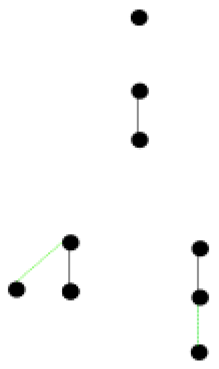
0) We start the linearisation at the root
1) Then, we select its dependent nodes [1,2,3] and randomly order them, which gives us [2,1,3].
2) We select their direction at random, which gives us ["left", "left", "right"], and the linearisation steps [0], [20], [120], [1203].
We repeat steps 1 through 2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

2) The second algorithm "original optimal" also sample 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 is inspired 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. The algorithm implementing this constraint is explained in the graph below :



0) We start the linearisation at the root
1) 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].
2) 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].
We repeat steps 1 through 2 until every node has been linearized, which gives us for example [425103]

3) 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 the graph below :



0) We start the generation process with a single node
1) We introduce a new node and randomly draws its governor. For now since there is only one potential governor, the edge has a probability of 1.
2) We introduce a new node and randomly draws 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.
3) We repeat this step until all nodes have been drawn and attached to their governor.[1]

These tree generation algorithms 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. For example during the linearisation process in 1) 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 weight.

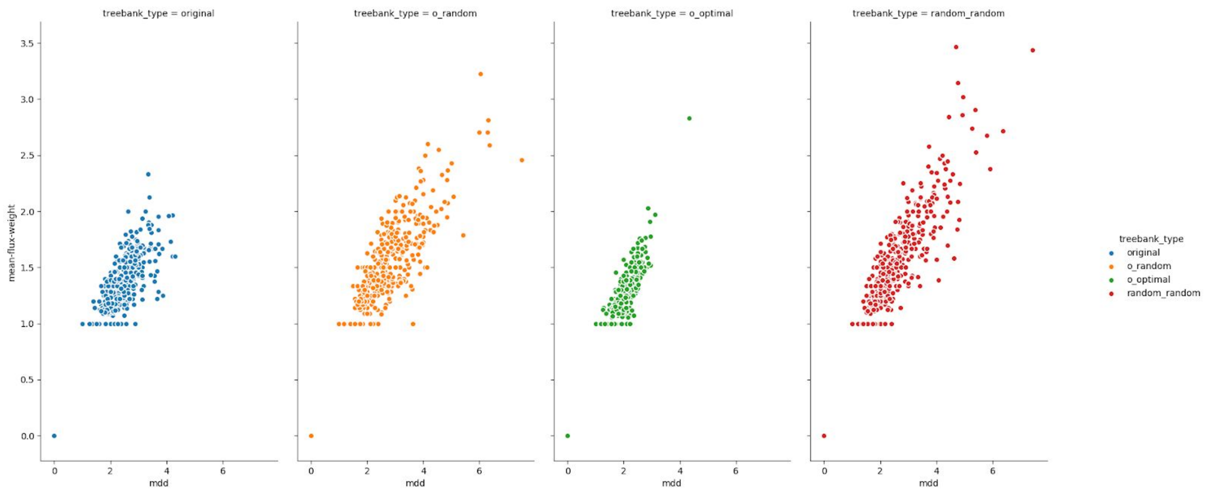
One question that still remains concerns the ordering of the two steps : unordered structure generation and linearisation generation. So far we have only implemented the full generation starting with the generation of the unordered structure and then moving on to the linearisation, which is a synthesis approach as described in Meaning-Text-Theory [Mel'čuk 1998], but it would be interesting to go in the analysis direction, where we start with a linearisation, that we then produce a structure for, in order to see if and how it affects the distributions of configurations inside the trees, especially as more constraints are added at different stages. We could then see if one type of random generation (synthesis vs analysis) produces structures that resemble natural dependency structures more.

Results and discussion

A continnum of constraints on structures

[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.

When we look at the relationship between different properties of the trees, we can see how the trees and their configurations are affected by the generation process. One such example can be found in the graph below :



This graph presents the relation between the mean dependency distance and the mean flux weight for every tree in the xxx treebank. Each facets illustrates a different type of manipulation on the trees. From the overall shape of the scatterplots, we can tell that the “original optimal” treebank has been generated with more constraints : it contains less variation in terms of configurations, which means that there is less dispersion and the relationship between the two properties is more apparent. The next treebank showing the most constraints is the original one, it has lost the “optimality” constraint, but is still more constrained than the two random treebanks as it has preserved the linguistic constraints of xxx (lang. The two random treebanks show more variation, with the “original random” still a bit less dispersed than its “random random” counterpart, as it still preserves the integrity of the unordered structure.

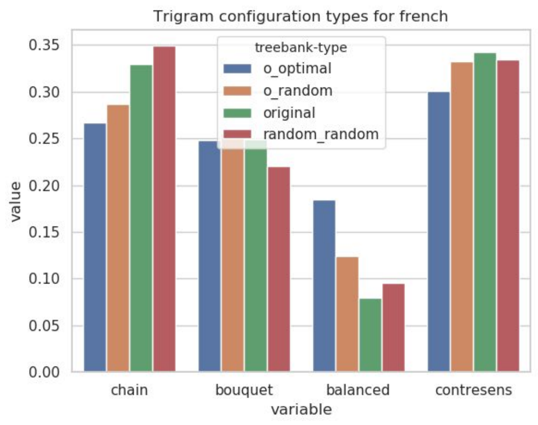
Correlation between properties

For each pair of properties, we measured the pearson correlation coefficient to find out how much the two variables are in a linear relationship. We looked into these results for the different natural treebanks and the artificial ones (“original random”, “random random” and “original optimal”). The full results are presented in tables 1-4 in annex.

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”). The correlation is the strongest in the artificial treebanks, especially in the “original optimal” version. On the other hand, the correlation between length and height is a bit stronger (xxxx means xxxx) in the original structures than in the random ones (“random random” is the the only format in which the height is affected by the manipulation). We also find quite strong correlations between mean dependency distance and height and mean dependency distance and length (xxxx all means xxxx). It is important to note that we find a lot of variation between different treebanks of the same language, which is why we also looked at the ranking of these correlations.

Trigram configurations

For configurations, we're mainly interested in the distribution of the four different groups presented in section (xxx hypotheses), and how they are impacted by language and the type of treebank (natural and artificial). We will discuss here a few points, and the full results are present in annex. In the plot below, we can see the distribution of these groups (in terms of percentage of all possible configurations, some of which are not in these four main groups) for french, depending on the type of treebank considered. In the original treebanks of french, the two main types of configurations are chains and contresens with 34% and 33% of all trigram configurations. The balanced strategy is far less used, as it represents only 7% of all occurences, which is interesting as this strategy is more frequent in the artificial treebanks, especially in those that optimize for dependency distances, where they reach 18%. The proportion of "bouquet" configuration is stable across the treebanks, with the exception of the completely random ones that use this strategy less. The two types of treebanks that include random components seem to favour chain and contresens configurations.



Across the four languages, the contresens configurations seem to take the lead, especially in japanese where they are far more frequent than in the artificial treebanks.

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 effects these constraints produce and how they interact with each other. We limited ourselves the artificial projective trees, which we now realize was a very strong constraint that restricts immensely the types of structures available, and therefore the variations of the different observed properties.

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. We also plan on digging deeper into the analysis of the present data, through the use of predicting models, that could help us clarify the relationship (whether they be linear or not) between the different features and to build a more solid basis to verify our hypotheses.

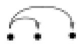
Références

Futrell, R., Mahowald, K., & Gibson, E. (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
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
Kahane, S., Yan, C., & Botalla, M.-A. (2017). *What are the limitations on the flux of syntactic dependencies? Evidence from UD treebanks*. 11.
Mel'cuk, Igor Aleksandrovic (1998). Dependency syntax: theory and practice. SUNY press.
Nivre, Joakim; Abrams, Mitchell; Agić, Željko; 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.
Temperley, D. (2008). Dependency-length minimization in natural and artificial languages*. *Journal of Quantitative Linguistics*, *15*(3), 256–282. https://doi.org/10.1080/09296170802159512

Annex

Types of configurations of trigrams

bouquet



chain

contresens

balanced

original

	depth_length	arity_length	arity_depth	mdd_length	mdd_depth	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)

o_optimal

	depth_length	arity_length	arity_depth	mdd_length	mdd_depth	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)

o_random

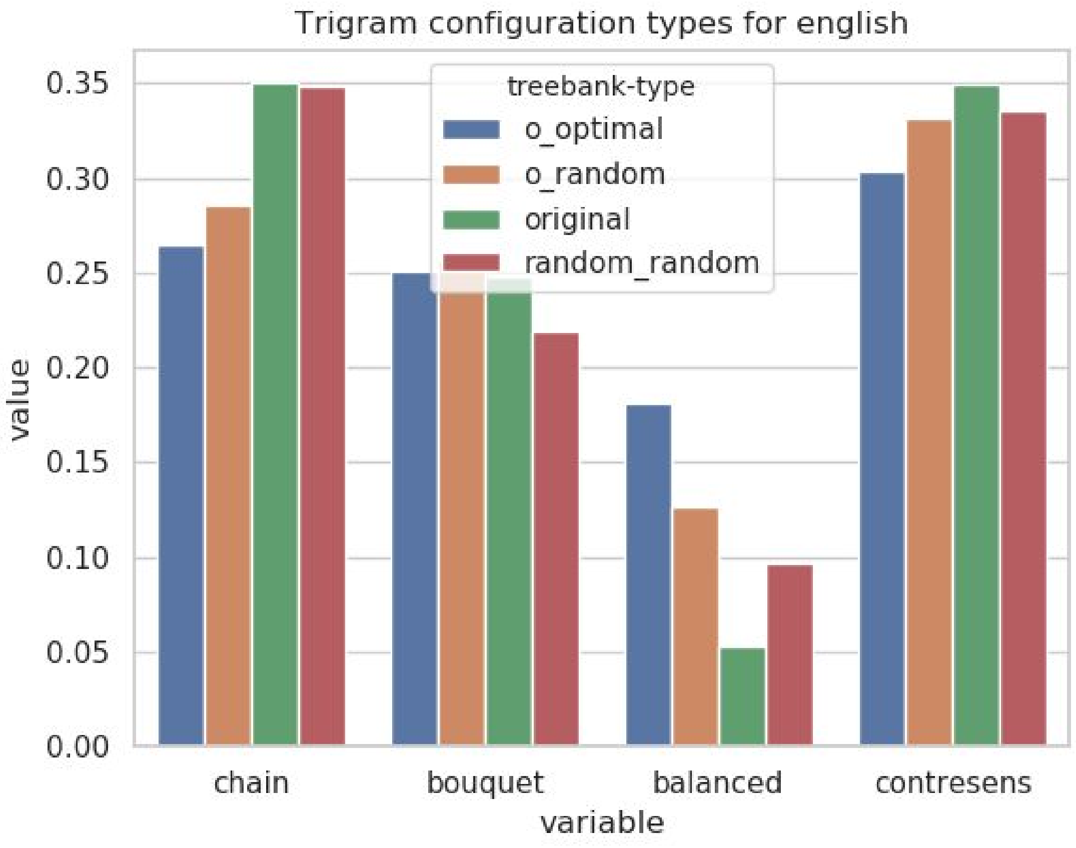
	depth_length	arity_length	arity_depth	mdd_length	mdd_depth	mdd_arity	mfw_length	mfw_height	mfw_arity	mfw_mdd
fr_pud	0.64 (3)	0.47 (7)	0.1 (10)	0.63 (4)	0.62 (5)	0.37 (8)	0.61 (6)	0.71 (2)	0.2 (9)	0.85 (1)
fr_spoken	0.84 (3)	0.59 (8)	0.42 (9)	0.82 (4)	0.8 (6)	0.6 (7)	0.81 (5)	0.86 (2)	0.4 (10)	0.9 (1)
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)

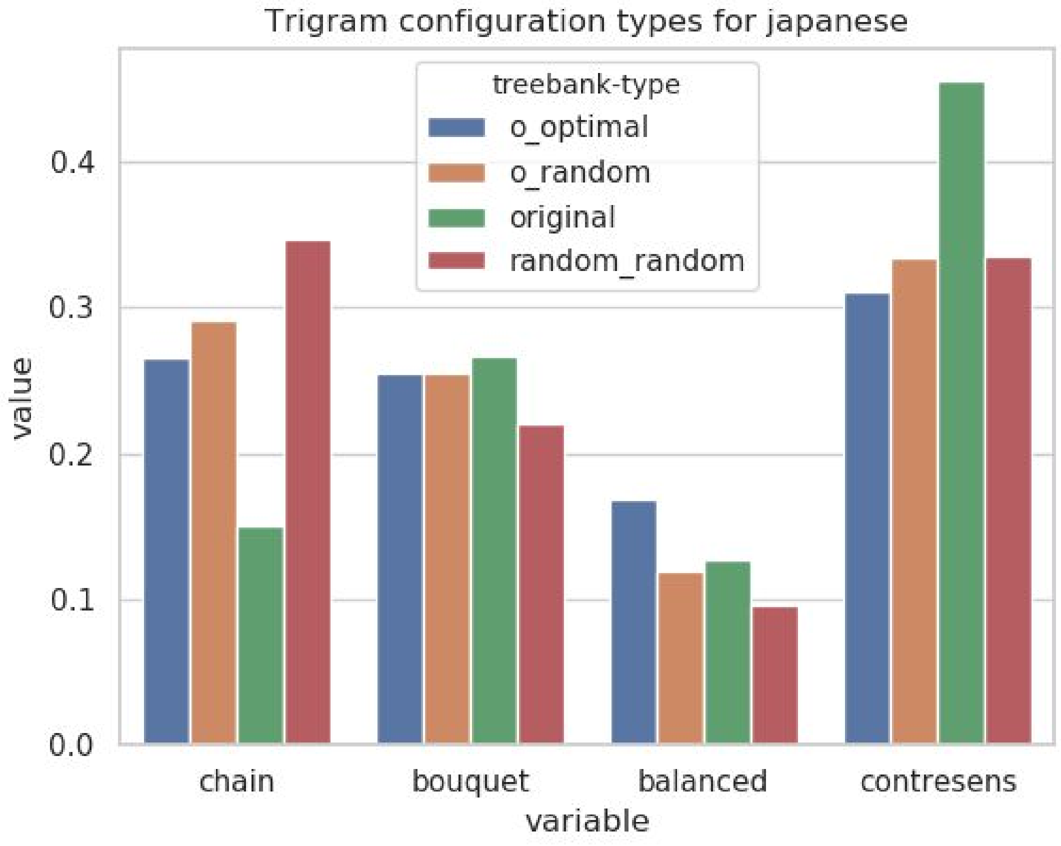
zh_cfl	0.76 (3)	0.61 (8)	0.37 (10)	0.75 (4)	0.67 (6)	0.63 (7)	0.71 (5)	0.76 (2)	0.42 (9)	0.85 (1)
zh_pud	0.57 (5)	0.53 (6)	0.07 (10)	0.63 (2)	0.47 (7)	0.45 (8)	0.59 (4)	0.62 (3)	0.18 (9)	0.81 (1)

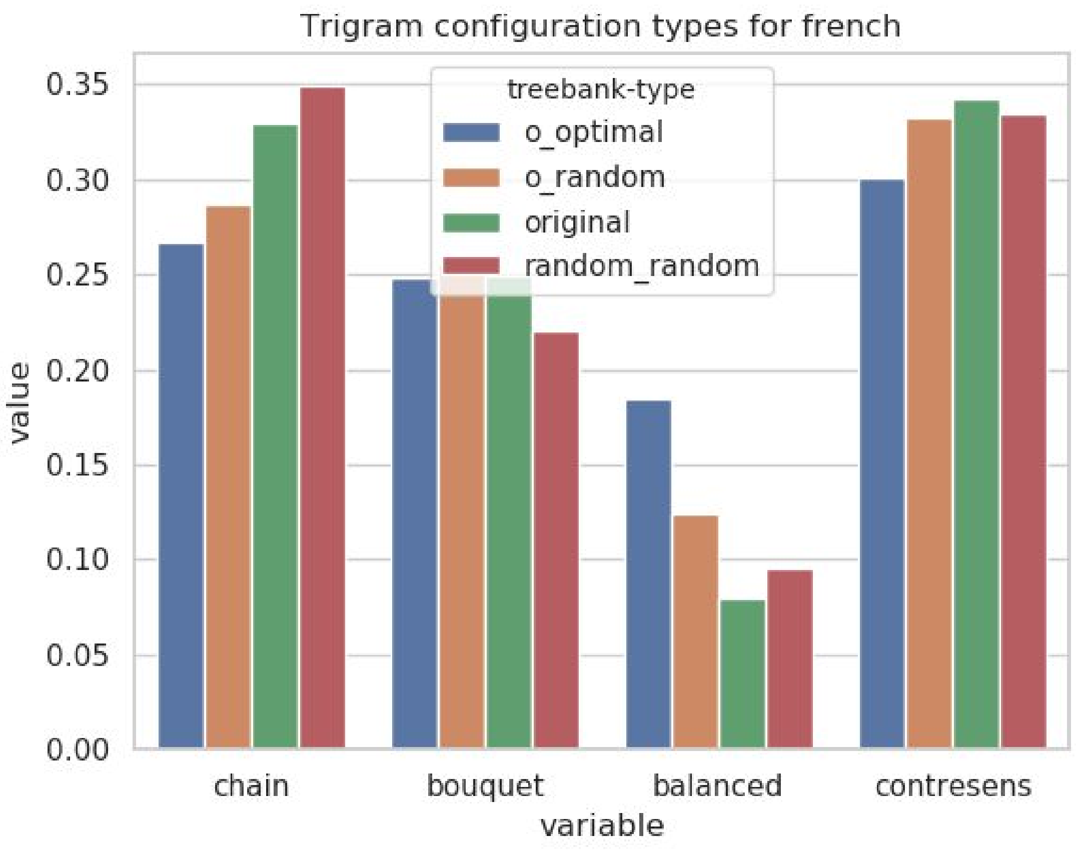
random_random

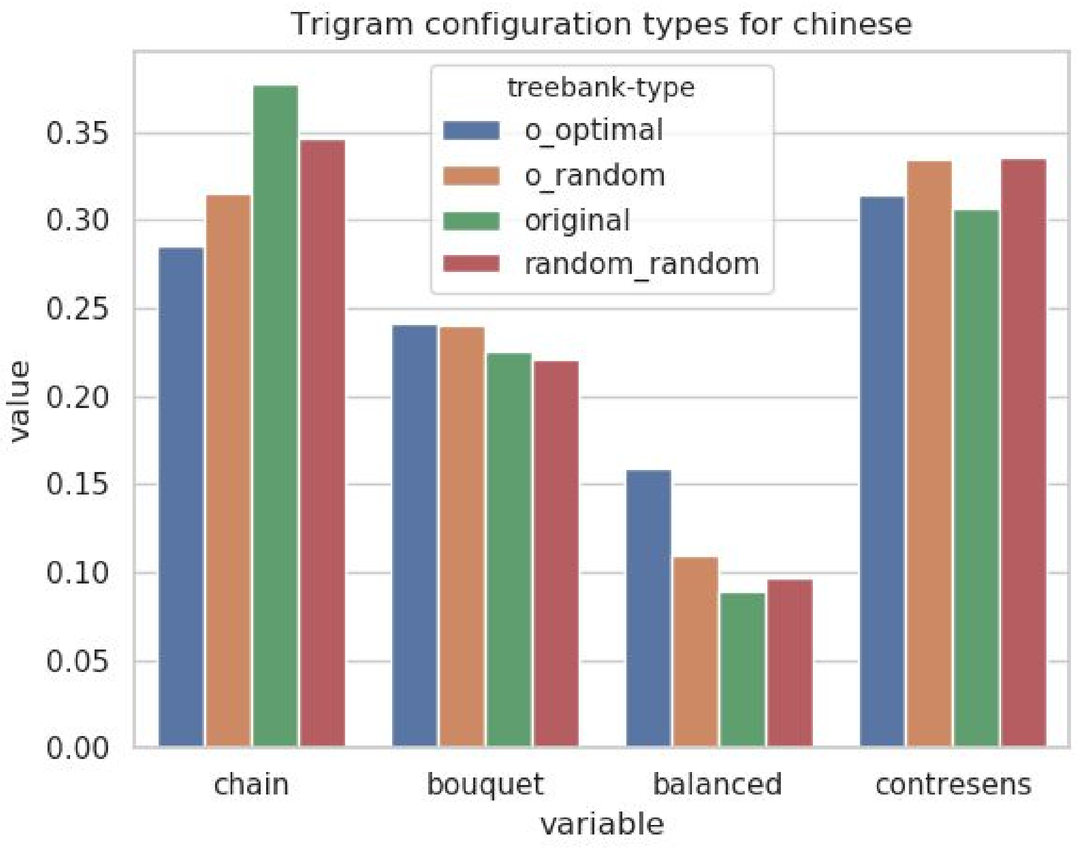
	depth_length	arity_length	arity_depth	mdd_length	mdd_depth	mdd_arity	mfw_length	mfw_height	mfw_arity	mfw_mdd
zh_gsd	0.61 (5)	0.53 (7)	0.14 (10)	0.65 (3)	0.55 (6)	0.42 (8)	0.64 (4)	0.65 (2)	0.23 (9)	0.85 (1)
zh_hk	0.8 (3)	0.75 (7)	0.58 (10)	0.8 (4)	0.79 (5)	0.73 (8)	0.75 (6)	0.81 (2)	0.63 (9)	0.91 (1)
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)

Trigram configurations by type across 4 languages


Trigram configuration types for english
treebank-type
o_optimal
o_random
original
random_random
value
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
chain
bouquet
balanced
contresens
variable


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
contresens
variable


Trigram configuration types for french
treebank-type
o_optimal
o_random
original
random_random
value
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
chain
bouquet
balanced
contresens
variable


Trigram configuration types for chinese
treebank-type
o_optimal
o_random
original
random_random
value
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
chain
bouquet
balanced
contresens
variable