Lab3: Significance of Network Metrics

Simon Van den Eynde
Martí Renedo Mirambell

October 27, 2016

1 Introduction

In this session we study the significance of the mean local clustering coefficient in syntactic dependency networks obtained from different languages. To test the significance of this metric, we will compare its values on our networks to those in two null models: the Erdös-Renyi graph and a random graph preserving the original degree sequence obtained with the switching model. We will estimate the corresponding p-values both by generating a large enough sample of each of the random networks and through analytical work.

2 Results

Table 4 shows how much time it takes to calculate the mean local clustering of a graph, the Erdős-Renyi graph and the switched graph. For every graph we calculated these mean local clusterings under different sortings: no sorting, increasing degree of vertex, decreasing degree, random sort, input sort. We did this for every language. And we did every calculation 5 times, to get a more accurate average. Notice that the mean local clustering was calculated here optimised.
We notice that for every language, except Chinese, sorting the graph according to increasingDegree results in a noticeable improvement in time.

Table 2 is calculated in the non-optimised way. We used $T = 20, Q = 10$, the time required is about 1010 seconds.

Table 3 is calculated in the optimised way. From the conclusions of table 4 we decided to sort every graph on increasing degree. We used $T = 30, Q = 15$.

Table 1: Summary of the properties of the degree sequences. N is the number of vertices of the network, E is the number of edges, $\langle k \rangle = 2E/N$ is the mean degree and $\delta = 2E/(N(N-1))$ is the network density of edges.

Language	N	E	$\langle k \rangle$	δ
Arabic	21532	68767	6,4	3,0e-04
Basque	12207	25558	4,2	3,4e-04
Catalan	36865	197318	10,7	2,9e-04
Chinese	40298	181081	9,0	2,2e-04
Czech	69303	257295	7,4	1,0e-03
English	29634	193186	13,0	4,4e-04
Greek	13283	43974	6,6	5,0e-04
Hungarian	36126	106716	5,9	1,6e-04
Italian	14726	56042	7,6	5,2e-04
Turkish	20409	45642	4,5	2,2e-04

Table 2: Values of the mean local clustering coefficient of the graph and of the Erdős-Renyi and switching models for $T = 20, Q = 10$

Language	C_{WS}	C_{WS}-ER	C_{WS}-switching
Arabic	0,188588	0,0003	0,1875
Basque	0,047010	0,0004	0,0536
Catalan	0,221312	0,0003	0,2291
Chinese	0,171235	0,0002	0,1613
Czech	0,121759	0,0001	0,1316
English	0,235362	0,0005	0,2404
Greek	0,133821	0,0005	0,1480
Hungarian	0,050899	0,0002	0,0931
Italian	0,144128	0,0006	0,1993
Turkish	0,223786	0,0003	0,2476

Table 3: Values of the mean local clustering coefficient and its p-values with respect to the binomial (Erdős-Renyi) and switching models for $T = 20, Q = 10$

Language	C_{WS}	p-value (binomial)	p-value (switching)
Arabic	0,19	0	0,2
Basque	0,05	0	1
Catalan	0,22	0	1
Chinese	0,17	0	0
Czech	0,12	0	1
English	0,24	0	1
Greek	0,13	0	1
Hungarian	0,05	0	1
Italian	0,14	0	1
Turkish	0,22	0	1

Table 4: Time required to calculate mean local clustering under different sortings of the graph. The values are in seconds

Language	normal	random	increasingDegree	input	decreasingDegree
Arabic	1,93	2,00	1,66	1,92	1,91
Basque	0,24	0,24	0,18	0,24	0,24
Catalan	7,92	8,29	4,02	7,74	7,96
Chinese	6,53	6,69	6,91	6,80	6,56
Czech	12,95	13,01	7,93	13,01	13,74
English	7,86	8,17	4,12	7,57	7,84
Greek	0,69	0,60	0,42	0,69	0,71
Hungarian	2,47	2,32	1,42	2,41	2,51
Italian	1,06	0,82	0,61	1,05	1,13
Turkish	2,36	2,53	1,29	2,52	2,55

3 Discussion

Erdös-Renyi As we can see in table 2, the results of the clustering coefficient of all languages are reasonably large for such sparse networks. In comparison, the corresponding Erdös-Renyi models have clustering coefficients of around 3 orders of magnitude smaller (both in the analytical and experimental cases). This, combined with the low variance of the metric (which is estimated analytically in 4.1) gives p-values of 0 for all languages with R's default precision (i.e. it is almost impossible to obtain a higher clustering coefficient generating a random Erdös-Renyi graph with the parameters of our networks). In our experiments (see table 3) we also notice that all p-values are 0. These results are as expected, since we knew from the theory classes that the Erdös-Renyi graph has a low clustering coefficient.

Switching We can see in table 2 that the mean local clustering values of the switching model are closer to the mean local clustering of the real graph. We notice from table 3 that only Chinese is considered large in comparison with his switching model. Arabic is on the border, but with a p-value of 0.2 we cannot say it has a large clustering. All the other languages have a p-value of 1 and therefore we cannot say their mean local clustering is high in comparison with the switching model (actually, in this case a p-value of 1 suggests that the C_{WS} of the real network is lower than its value in the switching model with statistical significance).

4 Methods

4.1 Analytical estimation of the p-values

4.1.1 Erdös-Rényi Graph

Given our original graph with N vertices and E edges, the Erdös-Rényi graph has the same size and order but with its edges randomized. Of the two possible Erdös-Renyi models, we will use $G(N,p)$ where p will be such that the expected number of edges is E (which is $p = \frac{E}{\binom{N}{2}}$). This has the advantage that X_j the random variables that for every possible vertex indicate wether it exists (X_j=1) or not (X_j=0) are independent, which will be very useful later on. When working with large values of N and E (such as those we study in this lab session), the models $G(N,E)$ and $G(N,p)$ should give very similar random graphs. For X_j where j is the index of any vertex, we calculate its expectation and variance:

$$\mathrm{E}[X_j] = 0 \cdot (1-p) + 1 \cdot p = p$$

$$\mathrm{Var}(X_j) = \mathrm{E}[X^2] - (\mathrm{E}[X])^2 = p - p^2$$

Given a vertex v of the graph, it can have $N-1$ neighbours (each with probability p), which results in $\binom{N-1}{2}$ possible pairs of neighbours. The expected number of pairs. This gives an expected number of pairs of neighbours $n_v = p^2\binom{N-1}{2}$. Then, we can estimate the local clustering of v by

$$C_v^{ER} \approx \frac{\sum_{j=1}^{\lfloor n_v \rfloor} X_j}{n}.$$

Since X_j are independent and equally distributed, we can apply the central limit theorem, which states that for a large enough sample size, the distribution of C_v^{ER} is the normal $N(\mathrm{E}(C_v^{ER}), \frac{\mathrm{Var}(X_j)}{n}) \approx N(C_v^S, \frac{p-p^2}{p\binom{n-1}{2}}) = N(C_v^S, \frac{1-p}{p\binom{N-1}{2}})$.

We can apply the central limit theorem again to $C_{WS}^{ER} = \frac{1}{N}\sum_{i=1}^{N} C_i$, wich gives us the distribution of C_{WS}^{ER}: the normal distribution

$$N(C_v^S, \frac{1-p}{pN\binom{N-1}{2}}) = N(C_v^S, \frac{1-p}{E(N-2)}).$$

Note that in the last step we used the equality $pN\binom{N-1}{2} = \frac{EN\binom{N-1}{2}}{\binom{N}{2}} = E(N-2)$.

Now the p-value of C_{WS}, which is given by $p(C_{WS}^{ER} \geq C_{WS})$, can be calculated easily in R (`analytical_pvalues.R`).

4.2 Empirical estimation of the p-values

The implementation of all the methods described in this section was donein Java and can be found in the folder `Lab3Java`.

Representation graph For the internal representation of the graph we used a linked hashmap with as keys the nodes and as values the set of neighbours. This allows us to add a sorting on the keys and have amortized constant acces times to the neighbours of a node. Also we keep the nodes in a hashset, so we can lookup if a vertex is contained in the set of neighbours in amortized constant time.

Mean local clustering We implemented two ways of calculating the mean local clustering: standard and optimised. Standard calculates for every vertex its contribution by checking every pair of neighbours. optimised does the same, but takes a mean local clustering value as parameter and checks if it:

- already reached that value: stop, return true;
- it is not possible to reach that value anymore, even if all remaining vertices return the maximum possible local clustering.

So in both cases we would not evaluate the last vertices. This immediately explains why sorting on increasing degree is so effective. We noted in previous lab session that the languages are power-law-alike modelled. This means that there are a few vertices with a very high amount of neighbours. Now if we put these at the end, the chances are great that we can skip these and as such save a lot of computation time.

Switching model For generating the switching model, we keep a list of all the directed edges (so half of the edges). We generate 4 random numbers, 2 to choose edges and 2 to choose a direction for the edges. The list only requires constant time to find the element and also constant time to set the new element.

So we get 4 vertices a, b, c, d for an edge $a \to b$ and an edge $c \to d$. We want to switch these to $a \to d$ and $c \to b$. To check if this possible we first check if any of the vertices is not unique, if only $a = c$ or $b = d$, we have no problem. Otherwise we cannot do the switching or we would create loops and we have to count the failure. We also check whether d is already a neighbour of a and whether b is already a neighbour of c. If this would be the case, we would create a multi-edge and since this is not allowed, we have to count a failure.

For calculating the p-value in the optimised way, we chose $Q = 15$ because it is then larger than the logarithm of the highest amount of edges (Czech language, $250000, \log(250000) = 12.5$). Because we do the amount of edges times Q switches (minus the failures), we will, according to the coupon collector problem, probably have switched every edge at least once.

Erdös-Renyi model We simply take two random vertices and check if they are already connected, if not add a connection. In this way we connect as much edges as we have to connect.

Sparse graphs Both the Erdös-Renyi model and the switching do not have many failures (switching model around one quarter, depending on the language). This is because the graphs we're dealing are sparse. Doing this analysis on graphs that are nearly complete would be a lot harder.