Word manifolds

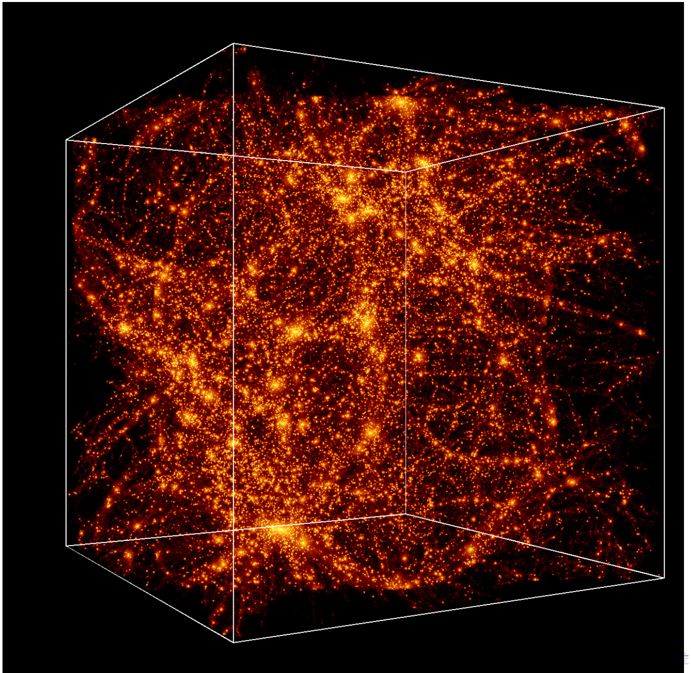
John Goldsmith and Wang Xiuli

May 26, 2015

Goals

- Visualize the global structure of a language
- Solve a technical problem in the unsupervised learning of morphology (past tenses of English verbs)
- Develop a language-independent method

16



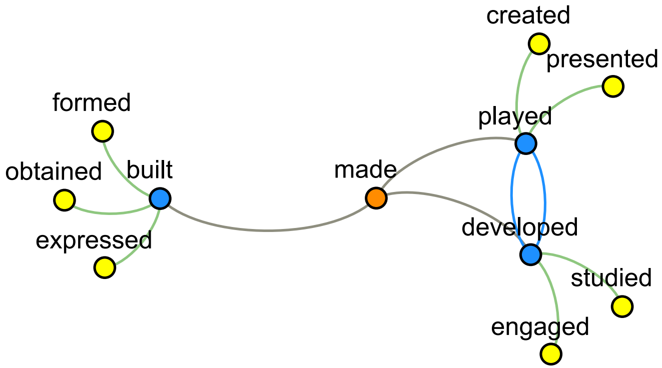
The algorithm is in three steps:

1. First, a determination of similarity between all pairs of words based on a simple comparison of word-context, and the creation of a graph $\mathcal{C}$ whose edge-weights is determined directly by those similarities.
2. Second, the computation of the K most significant eigenvectors of the normalized Laplacian of graph $\mathcal{C}$, and the calculation of the coordinates of each of the words in R^k based on these eigenvectors (where K is approximately 10).
3. Third, calculation of a new distance $d(.,.)$ between all pairs of words, viewing the words as points in R^K; a new graph $\mathcal{S}$ is constructed, whose edge weights are directly based on distance in R^K.
   The graph $\mathcal{S}$ can be directly viewed, using data visualization tools such as Gephi, and various clustering techniques can be applied to it as well.

First step: 1

Property			
W(-1)	=	w_j	the word to the immediately left of w is w_j
W(1)	=	w_j	the word to the immediately right of w is
W(-2)	=	w_j	the word two words left of w is w_j; etc.
W(-2,-1)	=	(w_j,w_k)	W(-2)=w_j and W(-1)=w_k.
W(-1,1)	=	(w_j,w_k)	W(-1)=w_j and W(1)=w_k.

‘*made*’ 3-neighbors and 2 generations



First step: 2

With all of our experiments described below, we have used the three features W(-2,-1), W(-1,1), and W(1,2).
Thus, in a corpus consisting *the sun shines in Hyde Park*, the word *shines* would be assigned three features: (*the, sun*); (*sun, in*); and (*in, Hyde*).

First step: 3

- Let V be the number of distinct word types in the language.
- Then there are in principle V features of the type W(-2,-1), and also of the type W(-1,1) and W(1,2).
- But the number of such features that are actually used is a small subset of the total number.
- For example, in an English-language encyclopedia composed of 888,000 distinct words, there were 1,689,000 distinct trigrams, of which 1,465,000 (nearly 87%) occur only once.

First step: 4

- We define $f(w_i, w_j)$ as the number of distinct features (using the contextual features just defined) shared by words w_i and w_j.
- It is natural to think of a graph $\mathcal{C}$in which the nodes are our words, and the edges are weighted by $f(w_i, w_j)$.
- The weight between two nodes indicates how many contexts they share, so all other things being equal, the stronger the weight of the edge between word A and word B, the more similar A and B are concerning their syntactic contexts.

Laplacian of a graph is a matrix

- The *laplacian* of a graph, such as $\mathcal{C}$, is defined as the matrix$M$ in which $M(i,j) = f(w_i, w_j)$ when $i \neq j$.
- For the diagonal elements, we first define $d(i)$ as $\sum_{k \neq i} M(i,k)$.
- $d(i)$ is the number of times word i appears in the corpus (you see that?).
- $M(i,i)$ is defined as $-1 \times d(i)$.

- We now have an initial similarity measure between words, but this similarity is not normalized for frequency: high frequency words will be much more similarity to others words that low frequency words will.
- Even if we normalize for frequency, though, the simplest ways of estimating similarity of distribution between two words on the basis of this data—using the cosine of the angle subtended by vectors pointing to each of the two words—is not as good as we might hope.

Second step: 1

- A number of researchers have explored the idea of taking a large set of data in a space of very high dimensionality, and finding a subspace of much lower dimensionality which is almost everywhere fairly close to the data.
- We've been especially influenced by the work of Partha Niyogi and Mikhail Belkin in the discussion that follows.

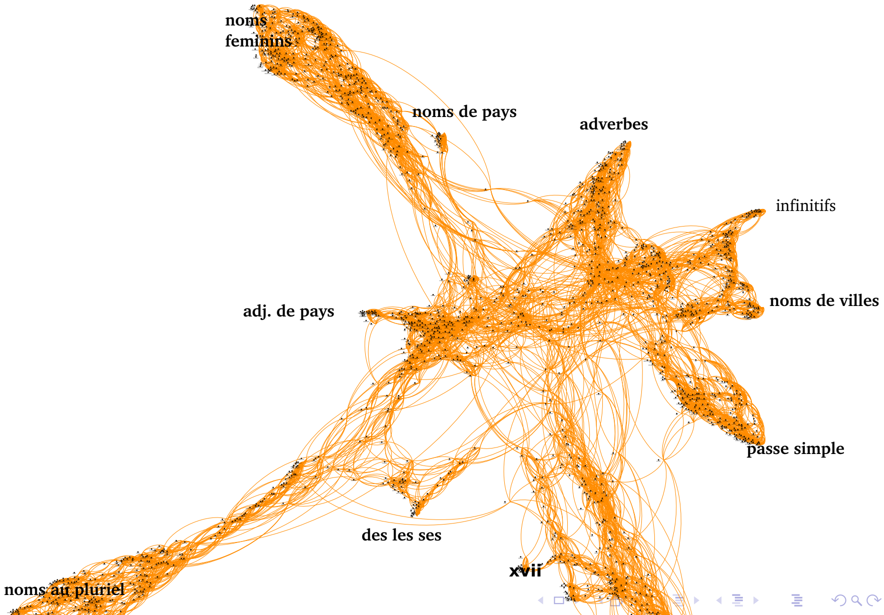
Second step: 2

- This means finding the eigenvectors of a normalized version of the graph laplacian.
- The normalized version of M, which we call N, is defined as follows: for all $i, N(i,i) = -1$, while for $(i,j), i \neq j$, we use the $d()$ function defined above to normalize, and say that $N(i,j) = \frac{M(i,j)}{\sqrt{d(i)d(j)}}$.

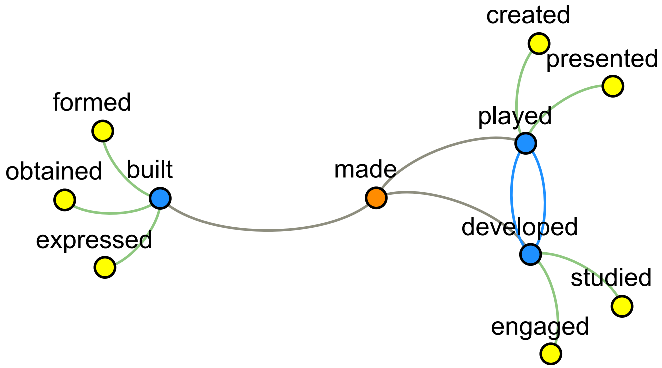
Second step and third step

We computed the first 11 eigenvectors of this normalized laplacian—those with the lowest eigenvalues, and used the 2nd through the 11th to give us coordinates for each word. Each word is thus associated with a point in R^{10}. We then select, for each word, the k closest words to it in this new space. These are the neighbors that we will explore below.

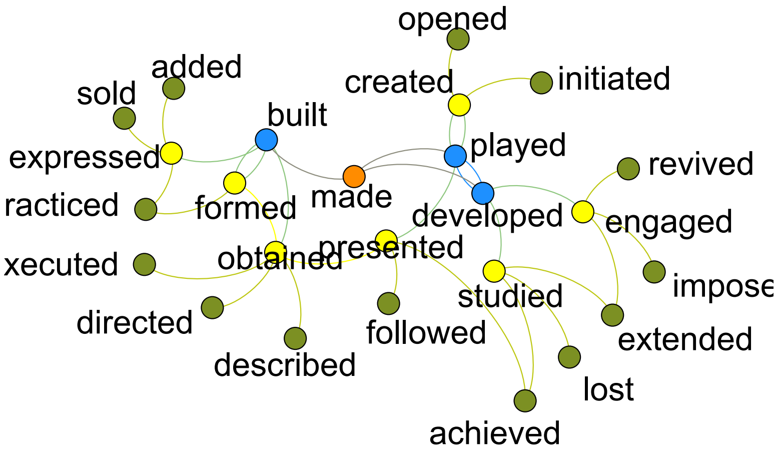
2,000 words of French



‘*made*’ 3-neighbors and 2 generations



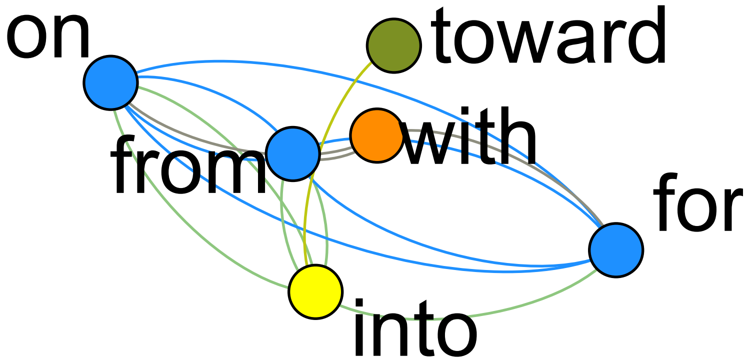
'*made*' 3 neighbors and 3 generations



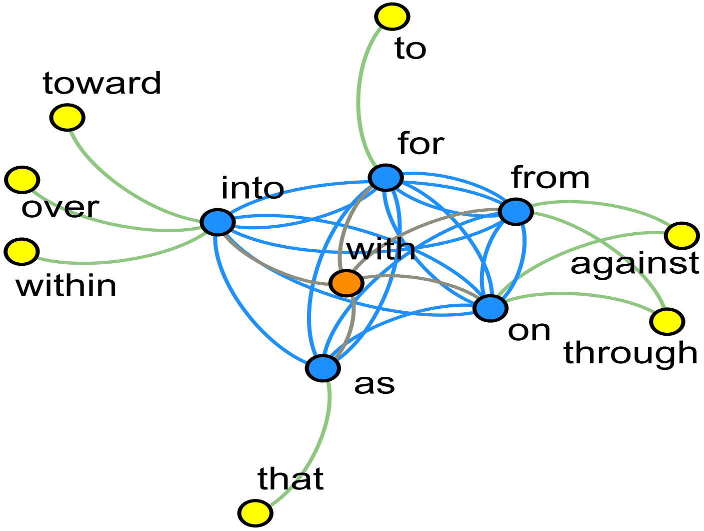
Help with learning morphology

jump	jumps	jumped	jumping	NULL-s-ed-ing
walk	walks	walked	walking	NULL-s-ed-ing
move	moves	moved	moving	e-es-ed-ing
build	builds	built	building	d-ds-t-ding
make	makes	??	making	NULL-s-ing

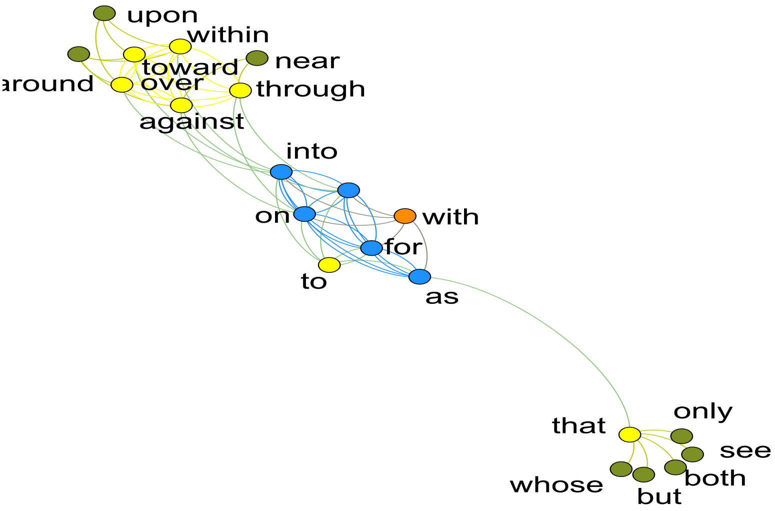
‘*with*’ 3 neighbors and 3 generations



‘*with*’ 5-neighbors and 2 generations



‘*with*’ 5-neighbors and 3 generations



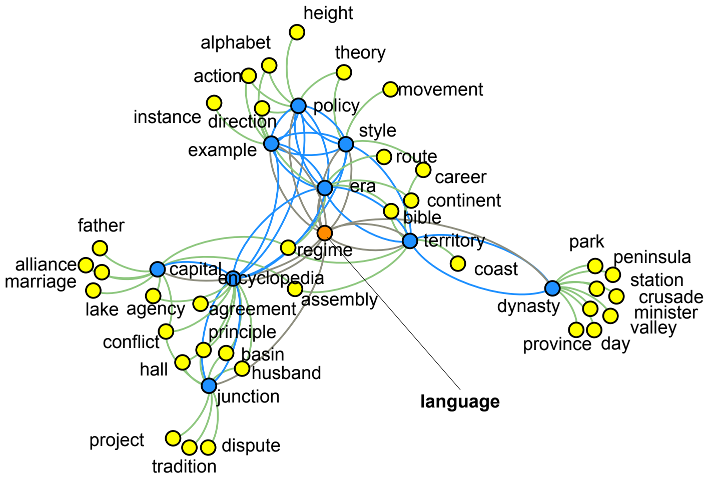
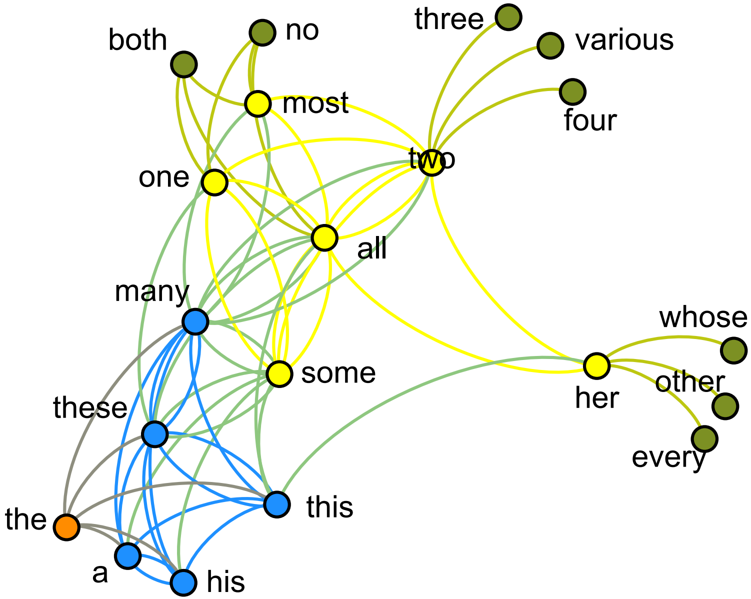
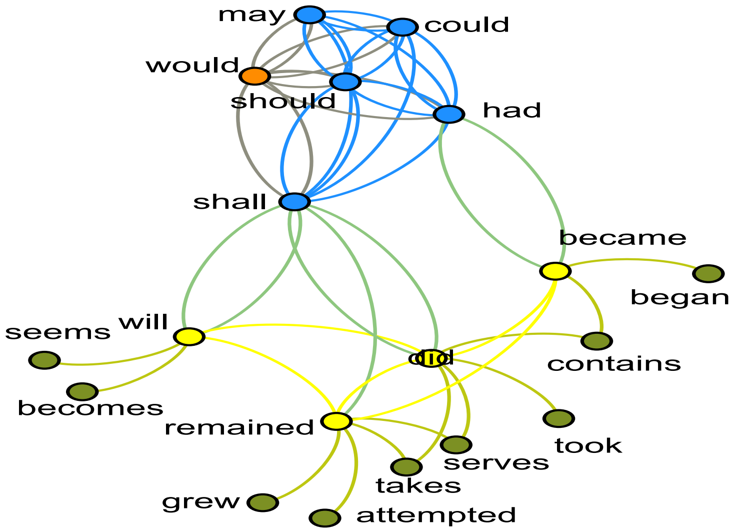


Figure : ‘*language*’ 9 neighbors and 2 generations

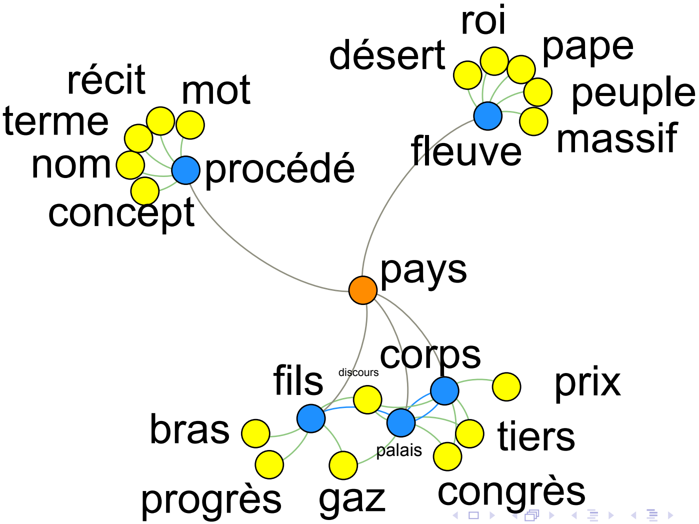
‘the’ 5 neighbors, 3 generations



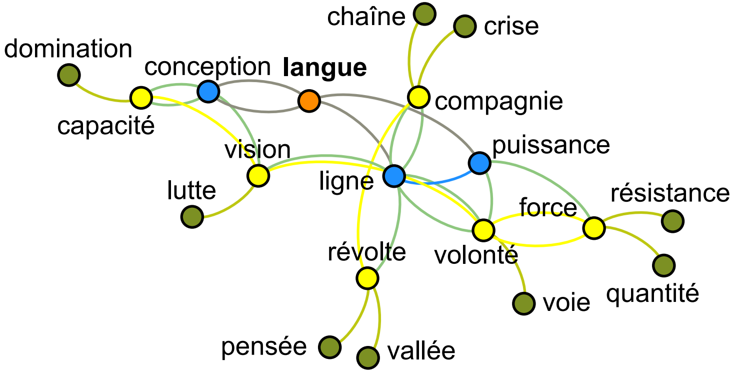
‘would’ 5 neighbors, 3 generations



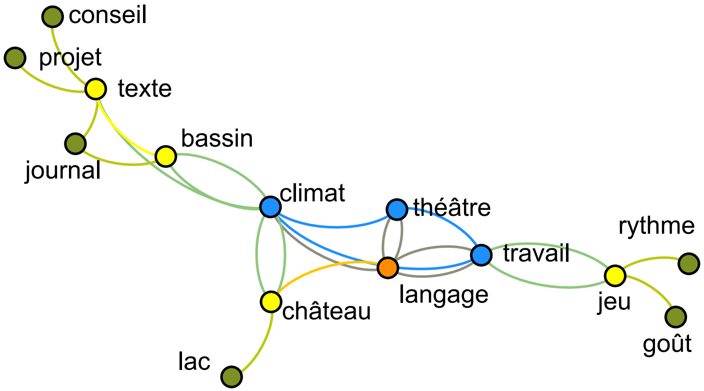
‘pays’ 5 neighbors and 2 generations



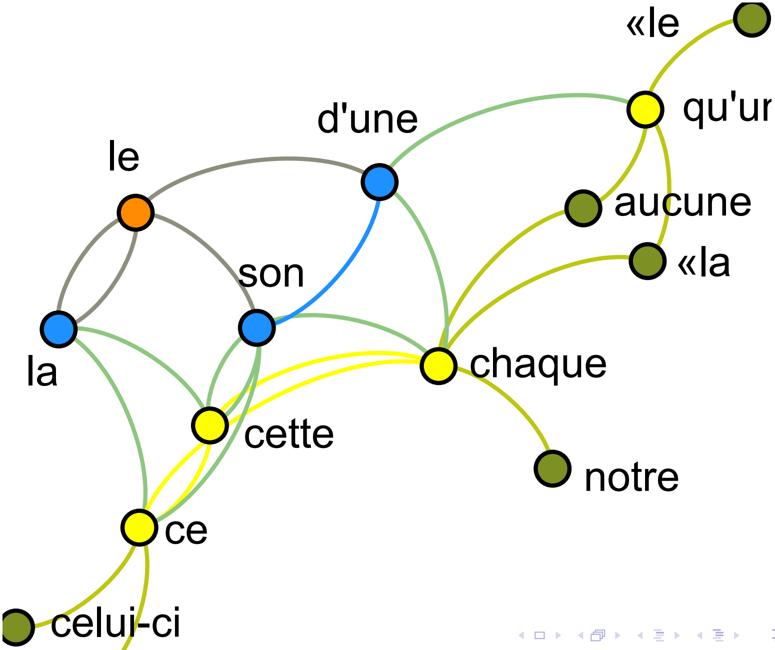
‘langue’ 3 neighbors and 3 generations



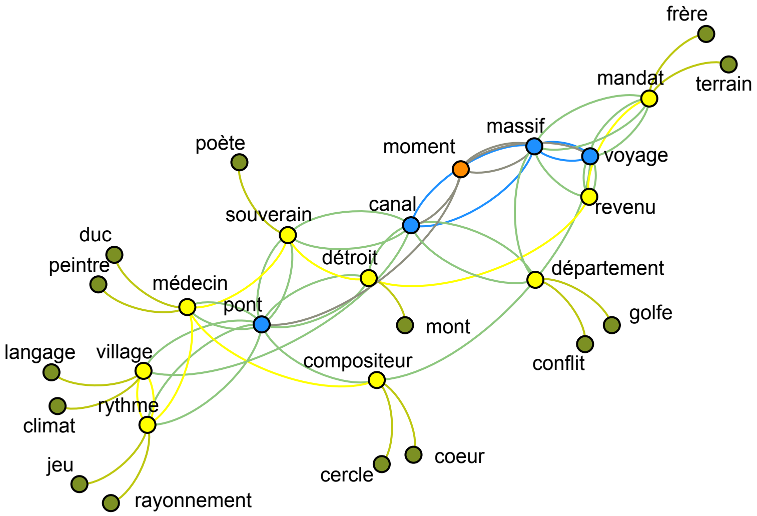
‘langage’ 3 neighbors and 3 generations



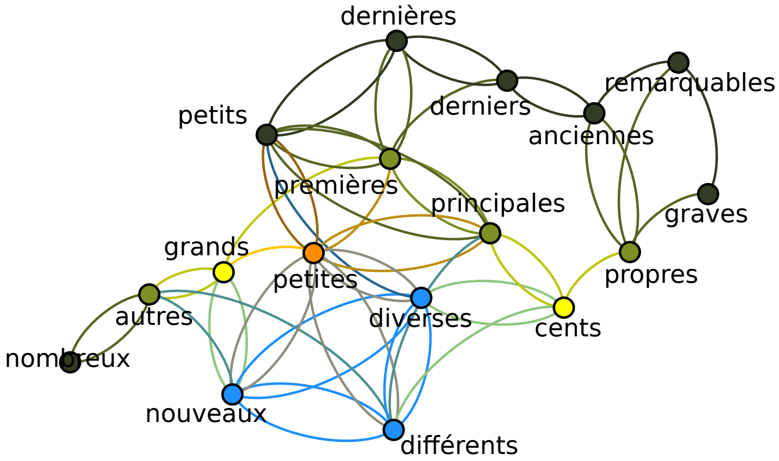
‘le’ 3 neighbors and 3 generations



‘moment’ 4 neighbors, 3 generations



petites, 3 neighbors



- There is a simple connection between minimizing the squared distance between nodes (though we haven't explained yet what kind of distance we are talking about now) of a weighted graph and the graph's Laplacian. We assume that no vertex is adjacent to itself.
- From a purely formal point of view, we could say that we are looking for a vector x in R^V which minimizes the expression, where W is the adjacency matrix of the graph, and $w_{i,j}$ are its entries:

$$\sum_{i,j} (x_i - x_j)^2 w_{i,j} \tag{1}$$

- Now we get to the kind of distance we're talking about: from the point of view of a projection, imagine that the entries $w_{i,j}$ in matrix W express the "similarity" between the i^{th} and the j^{th} element. We are looking for a single vector $\mathbf{x}$, then, which assigns very similar values to its i^{th} and j^{th} coordinate just in case those two coordinates correspond to elements that are "similar".
- We can think of that vector as representing a map from the graph's nodes to the real line; that is how we will think about it now, for the most part.

- We define a diagonal matrix D such that d_{ii} is the sum of the weights associated with edges adjacent to the i^{th} vertex: $d_{ii} = \sum_j w_{ij}$. Then
- $$\sum_i \sum_j (x_i - x_j)^2 w_{i,j} = \sum_i \sum_j (x_i^2 + x_j^2 - 2x_i x_j) w_{i,j} \qquad (2)$$

$$= \sum_i \sum_j (x_i^2) w_{i,j} + \sum_i \sum_j (x_j^2) w_{i,j} - 2 \sum_i \sum_j x_i x_j w_{i,j} \tag{3}$$

$$= \sum_i x_i^2 \sum_j w_{i,j} + \sum_j \sum_i (x_j^2) w_{i,j} - 2 \sum_i \sum_j x_i x_j w_{i,j} \tag{4}$$

$$= \sum_i x_i^2 d_{ii} + \sum_j x_j^2 \sum_i w_{i,j} - 2 \sum_i \sum_j x_i x_j w_{i,j} \tag{5}$$

$$= \sum_i x_i^2 d_{ii} + \sum_j x_j^2 d_{jj} - 2 \sum_i \sum_j x_i x_j w_{i,j} \tag{6}$$

- The first two terms are identical, and each are equal to $X^T DX$, while the third term is twice $X^T WX$. So

$$\sum_i \sum_j (x_i - x_j)^2 w_{i,j} = 2(X^T DX - X^T WX) = 2(X^T (D - W)X) \tag{7}$$

- It turns out that the matrix D-W has a name: it is the *laplacian* of the matrix W (or the graph of which W is the adjacency matrix). So we'll write $\mathcal{L} = D - W$. And there is a more natural way of writing $X^T (D - W)X$, which is to write $(X, \mathcal{L}X)$, which we can read as the inner product of the vector X and the vector $\mathcal{L}X$.

- If we restrict our attention to vectors of unit length, then this quantity $(X, \mathcal{L}X)$ is called the *Rayleigh quotient*. And we can find its maximal and minimal values along the eigenvectors of the laplacian. This is quite remarkable!
- Before we get to why that should be the case, we are going to squeeze the matrix so that its major diagonal consists of just 1's. We do this by defining a *normalized laplacian*, by dividing each entry l_{ij} of $\mathcal{L}$ by $\frac{1}{\sqrt{d_{ii}}\sqrt{d_{jj}}}$. We can write this:

$$\mathcal{L}' = D^{-\frac{1}{2}} \mathcal{L} D^{-\frac{1}{2}} \tag{8}$$

- If you are following this, you can see that $\mathcal{L}' = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$.
- The first term is the identity matrix; the second has 0s down the major diagonal, and is symmetric, and has only positive values; let's call it W', because it is the normalized form of W.
- And we have a better intuitive understanding of a matrix such as W', because it can naturally describe an ellipsoid: if we look at points x such that $(x, W'x)$ is a constant, we get an ellipsoid.
- Furthermore, $W'x$ is a vector normal to the surface of that ellipsoid at the point x.
- If we think about this geometrically, that means that $(x, W'x)$ will be a local maximum when x and $W'x$ point in the same direction — which is the same thing as saying that x is an eigenvector of W'.

- So we look at the eigenvectors of W', or of $\mathcal{L}'$. If we look at the eigenvectors of W', we sort them by decreasing eigenvalue, so λ_0 is the largest eigenvalue, and its eigenvector simply reflects the overall frequencies of the graph.
- Note: sometimes people start number the eigenvalues at 1, and sometimes at 0, as I have done here.] The second eigenvalue, λ_1, is of great importance in graph theory. Here we care about its eigenvector, though, and we look at the values it assigns to each word.