**Graphs**:

G(Vertices,Arcs)

V = all words

A = V^2

a=(I,j) is not a=(j,i)

Minimum(is the relation to the weights on the edges(in an intuitive sense, the tree with the lowest weights)) spanning (every node is in the tree(once, because otherwise not a tree)) tree(graph without cycles)

Use the minimum spanning tree algorithm, Chu-Liu-Edmonds (see page 20 of the chapter)

To make sure our root is our root, we have to modify our algorithm

HOW TO GET THOSE SCORES ON THE ARCS/EDGES?:

**NN**

We are going to assign scores (scores are positively framed => we will need to get the maximum spanning tree later on)

The main idea of a NN in language tasks is almost always:

* Take a discrete object like w1 = the and give it a meaningful ‘word embedding’ instead of for example: ‘this = word number 23’.
* So if we first give every word a unique ID. However than cat and cats can be very far apart while they are extremely similar. BAD CHOICE!
* If we go one step higher, we can assign a vector to it. For example [animal, plural, …,….] = [1,0,….] = cat. However it is hard to assign these numbers. Now the NN comes in.
* The NN will represent each word as a vector in R^d. You initialize the vectors randomly. How do you choose d? is a hyperparameter: 50-100 is usually used. Upside of a small d => less parameters, shorter training time, less data needed. Since we run it on our laptops, 50 is alright. These vectors are called the embeddings, you assign the word a place in the vectorspace.
* Can’t we use precomputed word embeddings? YES!
* Then train the NN to recognize similar words
* You can also give word embeddings to POS-tags. (all verbs close to each other etc.)
* LSTM’s are used as another layer on the word embeddings (figure 2 in 2017 paper). E.g. They give a representation of Kim in relation to other word in the sentence. This is important: if you have the word bank (ambiguous), you are adjusting the word bank based on all the words after and before(because it is a bi-lstm)
* To get the scores: you can take the innerproduct between the (normalized) vectors for ‘bank’ and ‘the’. The innerproduct coassigns the  
  However, innerproducts are commutative, both directions will get the same value. To solve this, make a version for both vectors that are a head and a dependent node (dependent is where the head points to). While doing this you put them both in a different space by using another perceptron: you apply a matrix and add a bias with a specific matrix and bias for dependency/head.
* So the figure 2 in the paper is just a fancy inner product.