Distributional Semantics

Lecture 8. From words to phrases, sentences and documents

Florian Gouret

May 4th, 2019

Presentation of the seminar

Plan:

1) Introduction
2) Sentence Embeddings and Document Embeddings
3) Multiword Expressions
4) Introduction to Syntax
5) Dependency-based Contexts. Word2VecF
6) Rhetorical structure theory
7) Evaluation
8) Some categories

Introduction and history

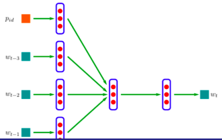
Sentence embedding, any idea?

Sentence Embeddings and Document Embeddings:

There exist data-driven approaches of word composition: one can consider a neural network as a composition function and train it to obtain the appropriate vector representations for phrases, sentences and tex.

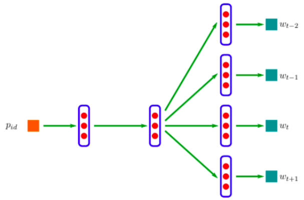
Sentence Embeddings and Document Embeddings: Paragraph vectors

In the Paragraph Vector framework, every paragraph is mapped to a unique vector, represented by a column in matrix D and every word is also mapped to a unique vector, represented by a column in matrix W.
The paragraph vector and word vectors are agregated (e.g. averaged or concatenated) to predict the next word in a context.
The paragraph token can be thought of as another word. It acts as a memory that remembers what is missing from the current context—or the topic of the paragraph. For this reason, this model is often called the **Distributed Memory Model of Paragraph Vectors (PV-DM)**.
the algorithm itself has two key stages: 1) training to get word vectors W, softmax weights U, b and paragraph vectors D on already seen paragraphs; and 2) *"the inference stage"* to get paragraph vectors D for new paragraphs (never seen before) by adding more columns in D and gradient descending on D while holding W, U, b fixed. We use D to make a prediction about some particular labels using a standard classifier, e.g., logistic regression.



Sentence Embeddings and Document Embeddings: Paragraph Vector Without Word Ordering: Distributed Bag of Words

The previous method considers the concatenation of the paragraph vector with the word vectors to predict the next word in a text window. Another way is to ignore the context words in the input, but force the model to predict words randomly sampled from the paragraph in the output. In reality, what this means is that at each iteration of stochastic gradient descent, we sample a text window, then sample a random word from the text window and form a classification task given the Paragraph Vector. This version is named the **Distributed Bag of Words version of Paragraph Vector (PV-DBOW)**.



Sentence Embeddings and Document Embeddings: Advantages of paragraph vectors

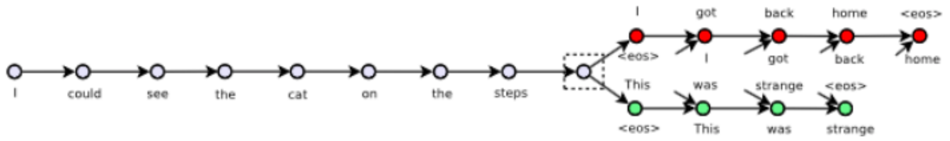
An important advantage of paragraph vectors is that they are learned from unlabeled data and thus can work well for tasks that do not have enough labeled data.
Paragraph vectors also address some of the key weaknesses of bag-of-words models.
The first, they inherit an important property of the word vectors: the semantics of the words.
The second advantage of the paragraph vectors is that they take into consideration the word order, at least in a small context, in the same way that an n-gram model with a large n would do. This is important, because the n-gram model preserves a lot of information of the paragraph, including the word order.

Skip-Thought Vectors:

Considering the following question: is there a task and a corresponding loss that will allow us to learn highly generic sentence representations?
Kiros et al. (2015), using word vector learning as inspiration, they propose an objective function that abstracts the skip-gram model to the sentence level. That is, instead of using a word to predict its surrounding context, we instead encode a sentence to predict the sentences around it. Thus, any composition operator can be substituted as a sentence encoder and only the objective function becomes modified.

Skip-Thought Vectors: Inducing Skip-thought Vectors

Skip-thoughts is considered in the framework of encoder-decoder models. That is, an encoder maps words to a sentence vector and a decoder is used to generate the surrounding sentences.
In this setting, an encoder is used to map e.g. an English sentence into a vector. The decoder then conditions on this vector to generate a translation for the source English sentence.



Skip-Thought Vectors: Inducing Skip-thought Vectors

Encoder Let $w_i^1, \dots, w_i^N$ be the words in sentence s_i where N is the number of words in the sentence. At each time step, the encoder produces a hidden state h_i^t which can be interpreted as the representation of the sequence $w_i^1, \dots, w_i^t$. The hidden state h_i^N thus represents the full sentence. To encode a sentence, we iterate the following sequence of equations:

$$\begin{aligned} r_i^t &= \sigma(W_r x_i^t + U_r h_i^{t-1}) \\ z_i^t &= \sigma(W_z x_i^t + U_z h_i^{t-1}) \\ \bar{h}_i^t &= \tanh(W x_i^t + U(r_i^t \odot h_i^{t-1})) \\ h_i^t &= (1 - z_i^t) \odot h_i^{t-1} + z_i^t \odot \bar{h}_i^t \end{aligned}$$

where $\bar{h}_i^t$ is the proposed state update at time t, z_i^t is the update gate, r_i^t is the reset gate $(\odot)$ denotes a component-wise product. Both update gates takes values between zero and one.

Skip-Thought Vectors: Inducing Skip-thought Vectors

Decoder The decoder is a neural language model which conditions on the encoder output h_i. The computation is similar to that of the encoder except we introduce matrices C_z, C_r and C that are used to bias the update gate, reset gate and hidden state computation by the sentence vector. One decoder is used for the next sentence s_{i+1} while a second decoder is used for the previous sentence s_{i-1}. Decoding involves iterating through the following sequence of equations:

$$
\begin{aligned}
r_{i+1}^t &= \sigma(W_r^d x_{i+1}^{t-1} + U_r^d h_{i+1}^{t-1} + C_r h_i) \\
z_{i+1}^t &= \sigma(W_z^d x_{i+1}^{t-1} + U_z^d h_{i+1}^{t-1} + C_z h_i) \\
\bar{h}_{i+1}^t &= \tanh(W^d x_{i+1}^{t-1} + U^d (r_{i+1}^t \odot h_{i+1}^{t-1}) + C h_i) \\
h_{i+1}^t &= (1 - z_{i+1}^t) \odot h_{i+1}^{t-1} + z^t \odot \bar{h}_{i+1}^t
\end{aligned}
$$

Multiword expressions:

Multiword expressions are expressions made of at least 2 words and which can be syntactically and/or semantically idiosyncratic in nature.
They are commonly used in any field of language – Jackendoff estimates the number of MWEs in a speaker's lexicon as comparable to the number of single words. Examples for MWEs would be idioms as „kick the bucket", compound nouns as „telephone box" and „post office", verb-particle constructions as „look sth. up" or proper names as „San Francisco". Due to the high frequency of MWEs there is a growing awareness in the NLP community for the problems they pose.

Multiword expressions:

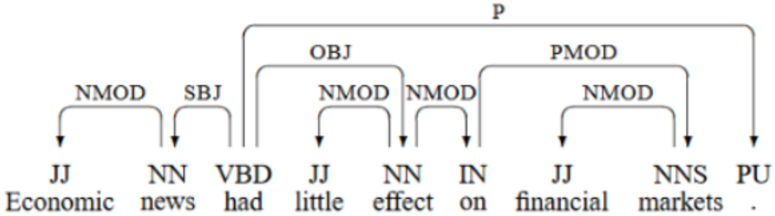
Fixed expressions are fully lexicalized and can neither be variated morphosyntactically nor modificated internally. Examples for fixed expressions are: *in short*, *by and large*, *every which way*.
non-decomposable idioms (i.e. idioms in which the meaning cannot be assigned to the parts of the MWE) such as *kick the bucket* the verb can be inflected according to a particular context: *he kicks the bucket*.
Proper Names are semi-fixed expressions as well since they can occur in different forms. For example the name of the U.S. sports team *the San Francisco 49ers* can occur as *the 49ers* or as a modifier in the compound noun a *49ers player*.
Syntactically-flexible expressions have a wider range of syntactic variability than semi-fixed expressions. They occur in the form of **decomposable idioms**, **verb-particle constructions** and **light verbs**.

Introduction to syntax:

Given a sequence of words, the element which allows people or machines to understand them and the information they are carrying is the **grammar**. It tells to the reader how to connect words within each other. The study of this fact is call the **syntax**.

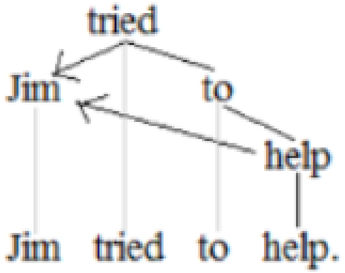
Introduction to syntax: Dependency grammar

The modern dependency grammar is a part of modern grammatical theories and was invented by Lucien Tesnière, a French linguist in 1959. The idea is that all words except one within a sentence depend on other words. The word without dependence is called the root.
Dependencies are motivated by grammatical functions; a word depends on another if it is a modifier or a complement.



Introduction to syntax: Dependency grammar

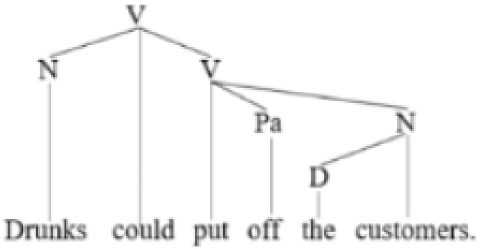
The tree representation can vary according to the representation model used for dependencies. For instance, the semantic representation considers dependencies in term of predicates and arguments. Each word of a sentence is linked to another one, order of words can be different than the one from the initial sentence.



Introduction to syntax: Dependency grammar

A constituent is a group of words within which there is a hierarchical structure. For instance, while reading and understanding a sentence, we tend to group certain words into some boxes such as subject, verb, object, this is these groups that constitute constituents. One method to analyze sentences comes from L.Bloomfield and is known as immediate constituent analysis. The constituent is traditionally represented under bracket:

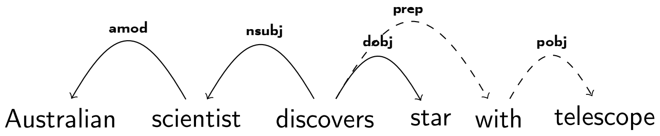
The blue dog can dance → [[[The] blue] dog] can [dance].

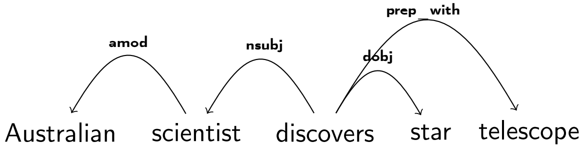


Dependency-based Contexts. Word2VecF:

In the Skip-gram embedding algorithm, the contexts of a word w_t are the words surrounding it in the text. The context vocabulary C is thus identical to the word vocabulary W. However, this restriction is not required by the model; contexts need not correspond to words, and the number of context-types can be substantially larger than the number of word-types. We generalize Skip-gram by replacing the bag-of-words contexts with arbitrary contexts. Syntactic contexts capture different information than bag-of-word contexts, as we demonstrate using the sentence *Australian scientist discovers star with telescope*.

Dependency-based Contexts. Word2VecF:



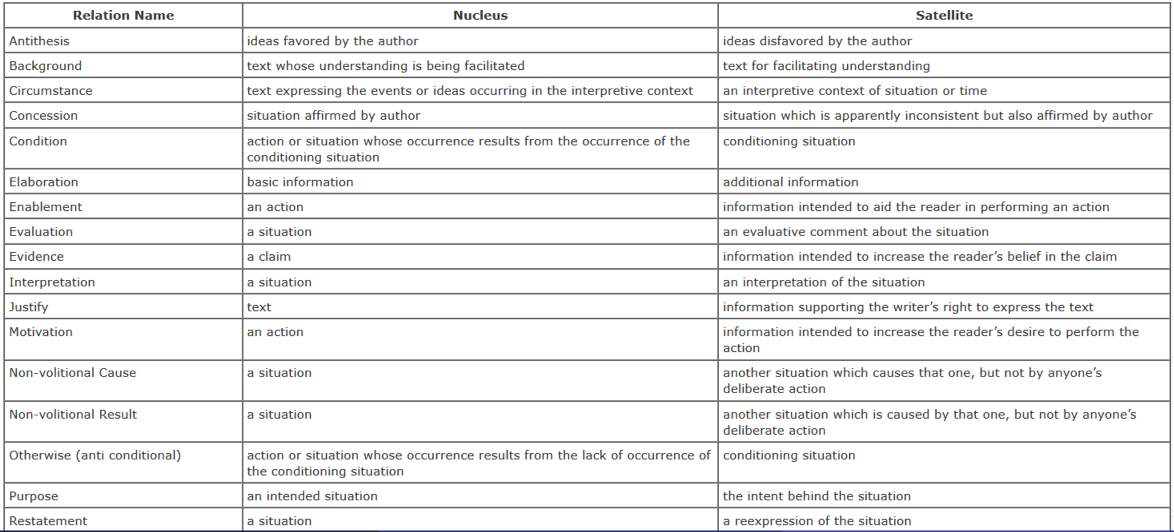


word	contexts
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

Rhetorical structure theory:

IIn 1983, Bill Mann, Sandy Thompson and Christian Matthiessen, working on automatic text generation highlighted the lack of information about discourse theory. Following the idea that texts are coherent; no gap during the retrieval of the information and the presences of clauses are justified, they came out with the rhetorical structure theory.
The rhetorical structure theory is a descriptive framework for text which used to investigate linguistic issues. It provides a combinations of features such as the meaning of relations, the grammar of clause combining... to describe relation among clauses. It can also be used to analyze narrative discourses or wide range of text.

Rhetorical structure theory:



Relation Name	Nucleus	Satellite
Antithesis	ideas favored by the author	ideas disfavored by the author
Background	text whose understanding is being facilitated	text for facilitating understanding
Circumstance	text expressing the events or ideas occurring in the interpretive context	an interpretive context of situation or time
Concession	situation affirmed by author	situation which is apparently inconsistent but also affirmed by author
Condition	action or situation whose occurrence results from the occurrence of the conditioning situation	conditioning situation
Elaboration	basic information	additional information
Enablement	an action	information intended to aid the reader in performing an action
Evaluation	a situation	an evaluative comment about the situation
Evidence	a claim	information intended to increase the reader's belief in the claim
Interpretation	a situation	an interpretation of the situation
Justify	text	information supporting the writer's right to express the text
Motivation	an action	information intended to increase the reader's desire to perform the action
Non-volitional Cause	a situation	another situation which causes that one, but not by anyone's deliberate action
Non-volitional Result	a situation	another situation which is caused by that one, but not by anyone's deliberate action
Otherwise (anti conditional)	action or situation whose occurrence results from the lack of occurrence of the conditioning situation	conditioning situation
Purpose	an intended situation	the intent behind the situation
Restatement	a situation	a reexpression of the situation

Evaluation of sentence embeddings:

As with all models created, the task of evaluating results and performances has to be done. For sentences, different possibilities are available according the range of application. The main difference compare with the evaluation of word embeddings is that there is no limits of uses, the number of sentences is not fix and the way they are made make them unique. Also finding way to evaluate sentence embeddings is more complex. However different categories may be taken such as paraphrase detection, entailment... leading to the possibility to create models.
In this logic to provide a method to evaluate sentence embeddings Alexis Conneau and Douwe Kiela present a model SentEval which aims to evaluate the quality of universal sentence representations in a centralized and less cumbersome way. Five different evaluations are presented such as entailment and semantic relatedness, paraphrase detection, semantic text similarity and caption-image retrieval.

Type theory:

Words about type theory and categorical grammar.

Type theory: Lambek calculus

For J. Lambek, the grammar can be modeled with some mathematic elements. Hence, he started to describe a way to do so.
The main idea is that words used in the same way should share the same structure.
By understanding the reason and the position of each, Lambek set up a multiple kinds of types allowing the grammar representation of sentences. However, it does not consider the semantic value; correct grammar sentences may have no semantic sense.

$$(x/y)y \to x, \quad y(y\backslash x) \to x \tag{1}$$

Type theory: Lambek calculus

he sees her

$$\pi_3 \left(\pi_3^r \mathbf{s}_1 \mathbf{0}^\ell\right) \mathbf{o} \rightarrow \mathbf{s}_1$$

Will he go with her?

$$\left(\mathbf{q}_1 \mathbf{j}^\ell \pi^\ell\right) \pi_3 \mathbf{i} \left(\mathbf{j}^r \mathbf{i} \mathbf{o}^\ell\right) \mathbf{o} \rightarrow \mathbf{q}_j \mathbf{j}^\ell \mathbf{i} \mathbf{j}^r \mathbf{i} \rightarrow \mathbf{q}_1 \mathbf{j}^\ell \mathbf{i} \rightarrow \mathbf{q}_1$$

Type theory: Lambek calculus

Within all types, two can be used to describe all others: n, the type of name and s, the type of sentence.

$$\textit{John works}$$

$$n \quad n \setminus s$$

$$\textit{John works and Mary rests}$$

$$n \quad n\backslash s \quad (s\backslash s)/s \quad n \quad n\backslash s$$

Type theory: Lambek calculus

Within all types, two can be used to describe all others: n, the type of name and s, the type of sentence.

$$\begin{array}{cc} \textit{John} & \textit{works} \\ n & n \setminus s \end{array}$$

$$\begin{array}{ccccc} \textit{John} & \textit{works} & \textit{and} & \textit{Mary} & \textit{rests} \\ n & n\backslash s & (s\backslash s)/s & n & n\backslash s \end{array}$$

Syntactic categories

Two types of syntactic categories are defined:

- t is the type for sentence
- e is the type for noun

If A and B are two syntactic categories, then A/B and $A//B$ are too.

Category	Categorial definition	Nearest transformation
e		None
t		None
IV	t/e	Verb phrase and intransitive verb
T	t/IV	Noun phrase and Proper name
TV	IV/T	Transitive verb
...		

Question

Questions?