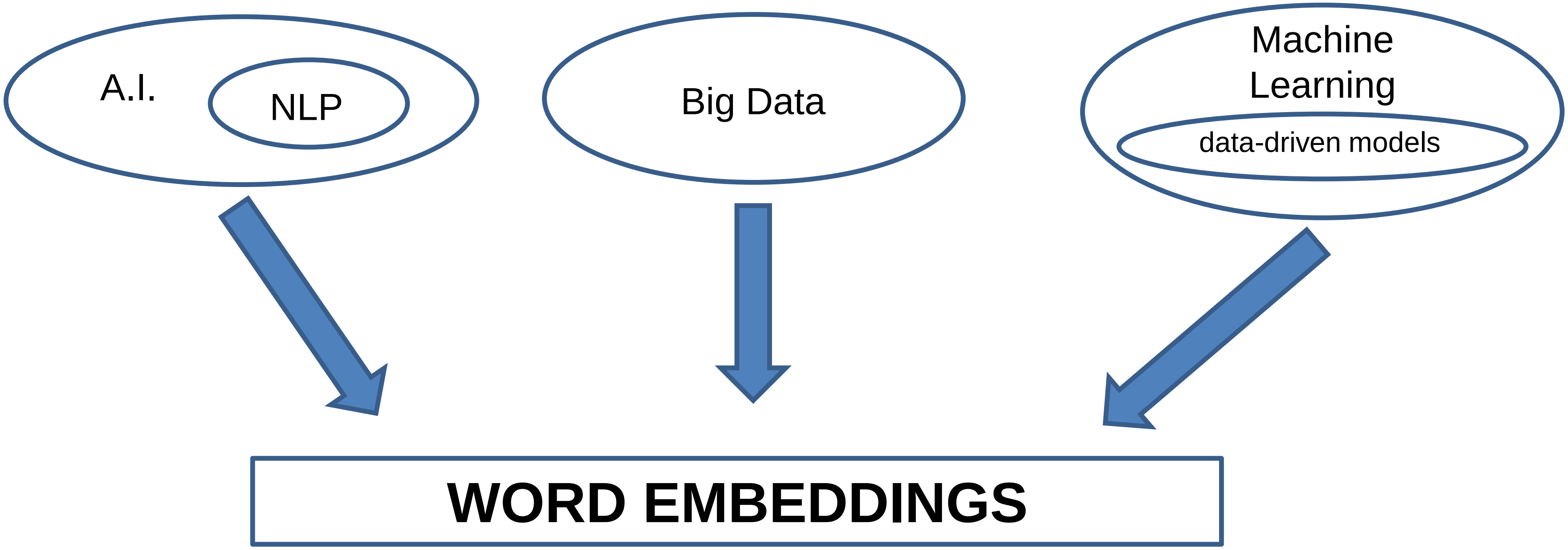
Word Embeddings models: theory and representation

Massimo De Cubellis - ISTAT

Reference framework



Definitions (1/2)

The **artificial intelligence (AI)** is a discipline belonging to the computer science that studies the theories, the methodologies and the techniques that allow to design hardware and software systems, capable of performing tasks typically exclusive of the human intelligence.

Among the main task that A.I. is able to perform, in addition to image / sound recognition or resolution of computational patterns, there is also the natural language processing

Natural language processing, also known as **NLP**, is the automatic elaboration process of information expressed in a natural language.

Definitions (2/2)

In statistics and computer science the term **big data** generically indicates a data collection so extensive in terms of volume, velocity and variety to use specific technologies and analytical methods for the extraction of value or knowledge (Wikipedia)

Machine learning is exactly one of these methods for extracting value and knowledge from (big) data; this technique foresee a learning phase in which a neural network try to identify patterns in the data, through an algorithm that improves the performance at each iteration

Data driven models: they are models in which there aren’t pre-defined algorithms or programs that “teach” the computer how to solve a problem

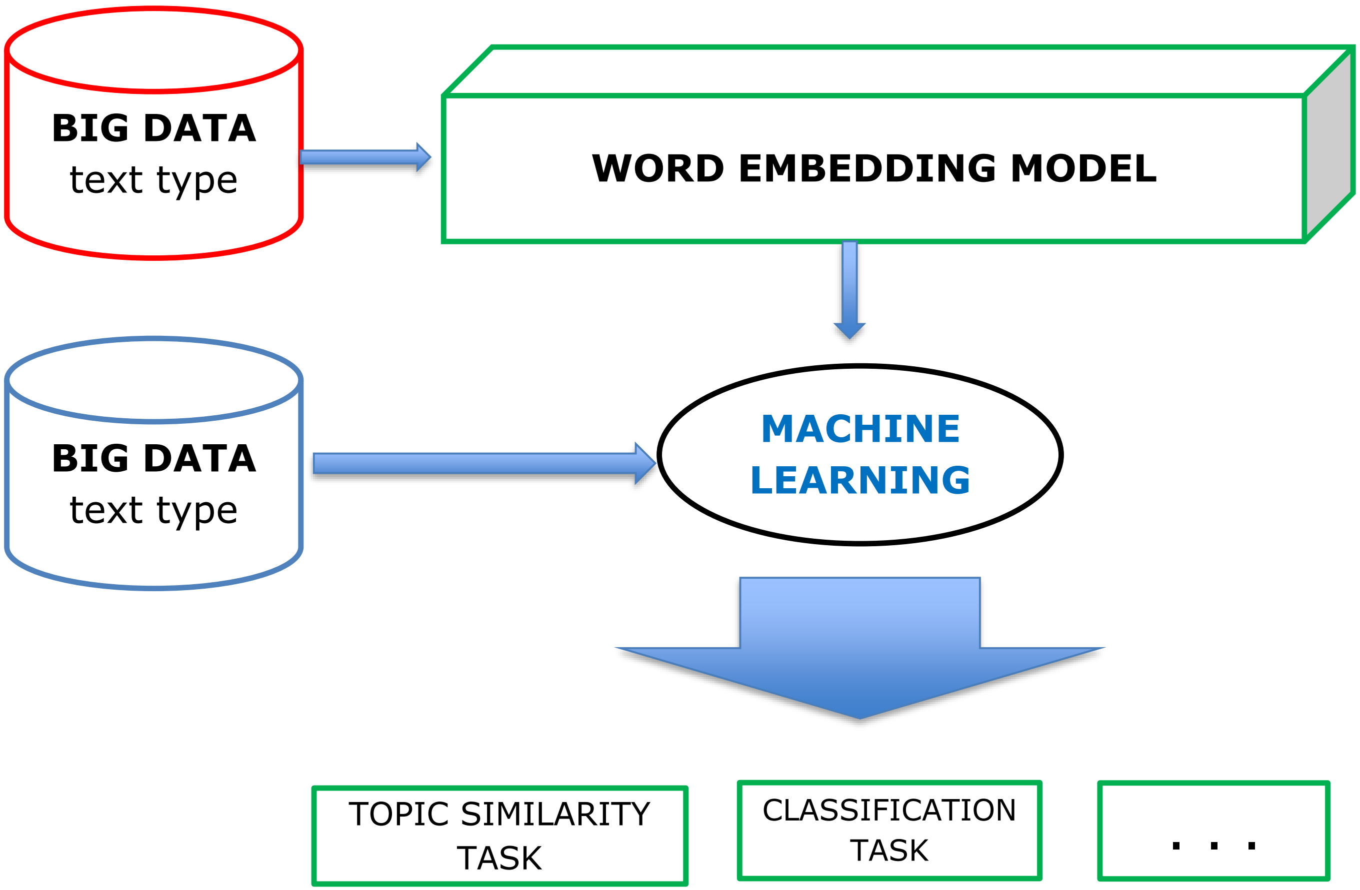
NLP's Applications

Some examples:

- speech recognition and synthesis
- automatic translation
- chatbot (software to simulate conversations with humans)
- summarization systems
- systems to solve topic similarity and text classification tasks
- sentyment analysis systems

And Word Embedding ?

NLP: from Big Data to Statistics tasks



Word Embeddings

Word Embedding models (1/2)

Word Embedding models map words in vectors of a *vector space*

The underlying hypothesis is that similar words appear in similar *contexts*:

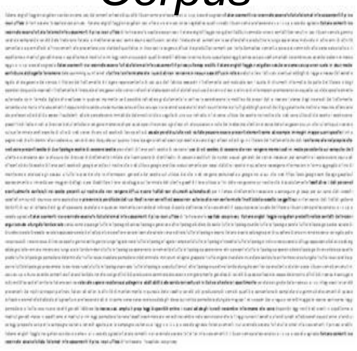
- *“You shall know a word by the company it keeps”*
- *‘the cat meows’ – ‘the kitten meows’*

Models adopted:

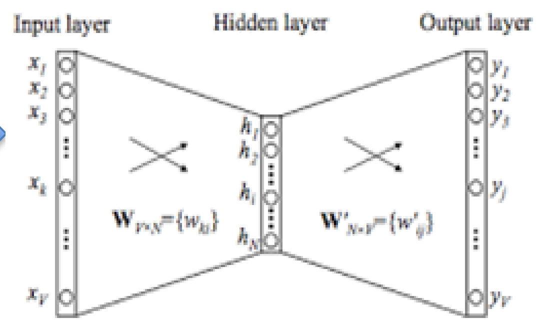
- Word2Vec (Google – 2013 - *Tomas Mikolov*)
- GloVe (Stanford University 2014)
- Fast Text – (Facebook Research - 2016)

Word Embedding models are DATA-DRIVEN

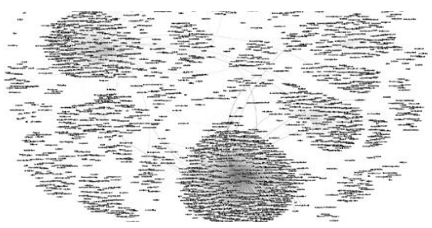
Corpus



Word2Vec



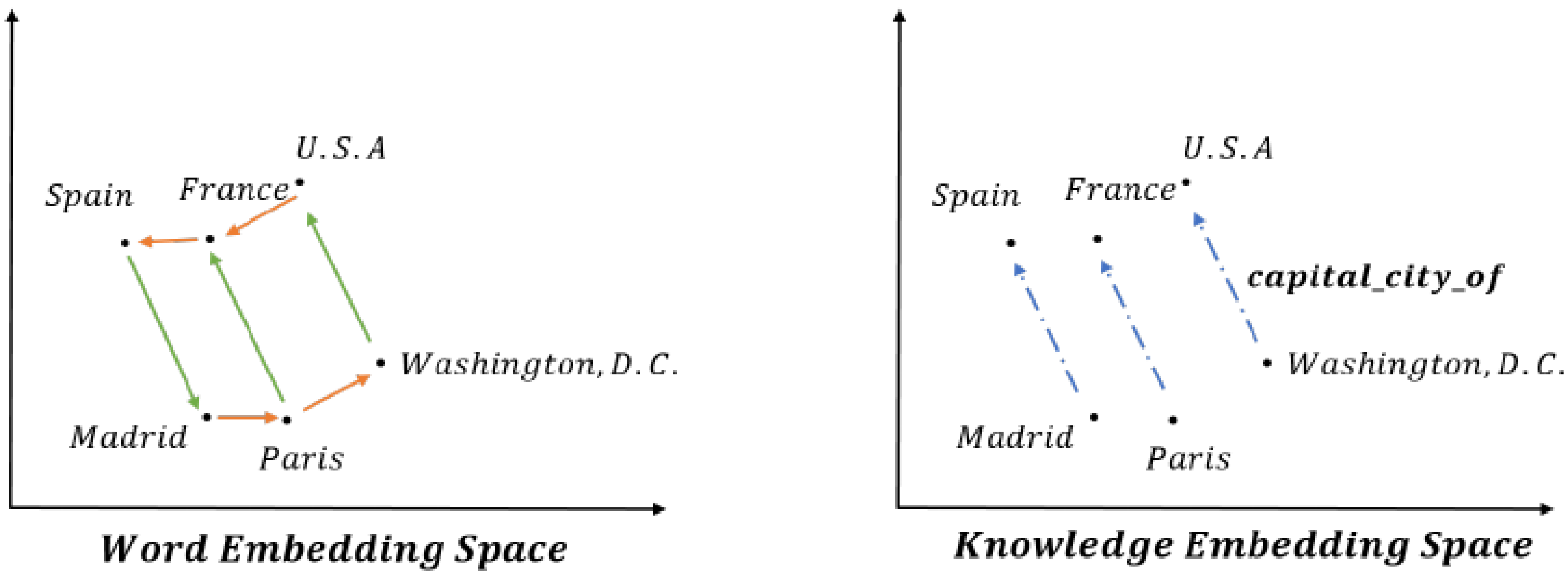
WE Model



Word Embedding models (2/2)

The first results obtained by Word2Vec (2013) were really amazing:

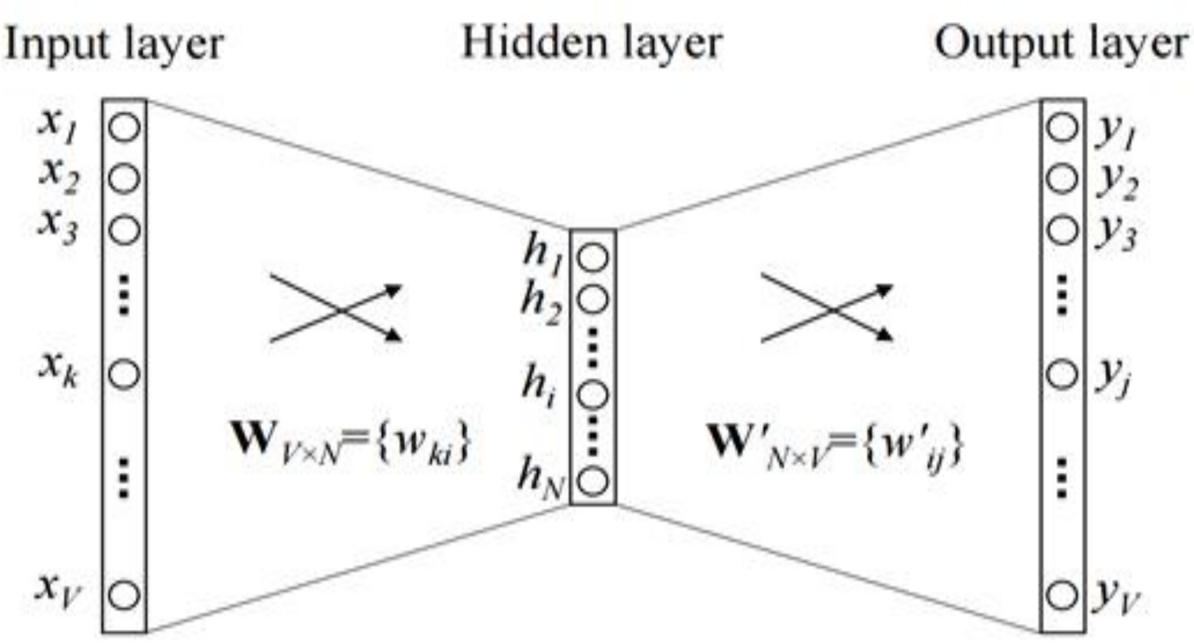
- words which are close, from a syntactic or semantic point of view, are represented by (almost) parallel vectors (*cosine distance metric*)



Word2Vec

Word2Vec: a simple neural network

Word2Vec is a simple not-supervised neural network with a single hidden layer



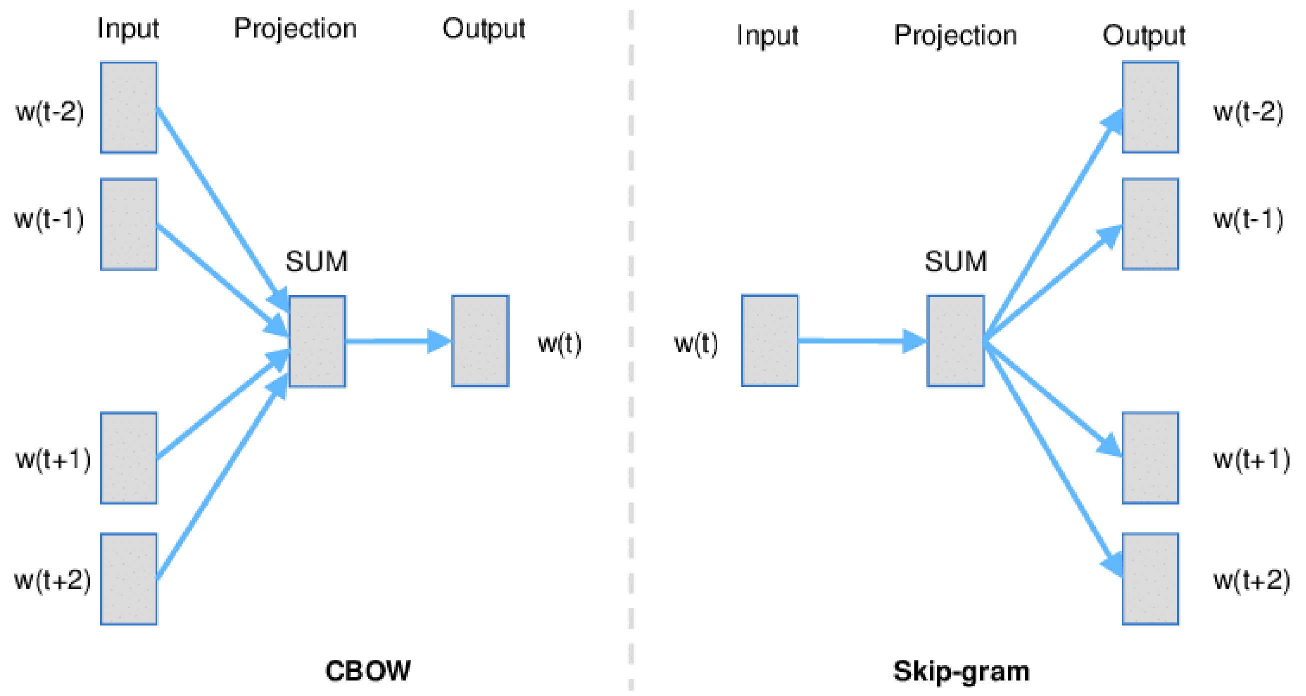
Context and sliding window

Sentences	Training pairs
the man who passes the sentence should swing the sword	(the, man) (the, who)
the man who passes the sentence should swing the sword	(man,the) (man,who (man,passes)
the man who passes the sentence should swing the sword	(who, the) (who, man) (who, passes) (who,the)
the man who passes the sentence should swing the sword	(passes, man) (passes, who) (passes, the) (passes, sentence)

Word2Vec: CBow vs Skip-gram

Word2Vec, during the training phase, can work in two ways:

- **predicting** the **central word** given its context (**Cbow**)
- **predicting** the **context** given its central word (**Skip-gram**)



Source: "Efficient Estimation of Word Representations in Vector Space" - Authors: Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean

Word2Vec: The hyperparameters

The main hyperparameters used by Word2Vec are:

- **embedding space dimension**: is the dimension of the vector space used to map the words of the corpus (e.g. 300)

- **window size**: is the width of the window used to slide the corpus. It defines how large the context is.

- **Cbow / SkipGram**: is the algorithm chosen to train the neural network

- **iteration**: is the number of time, during the whole process of training, in which the weights of the neural network are re-calculated

Neural Network Structure of Skip-Gram

Source Text

Training Samples

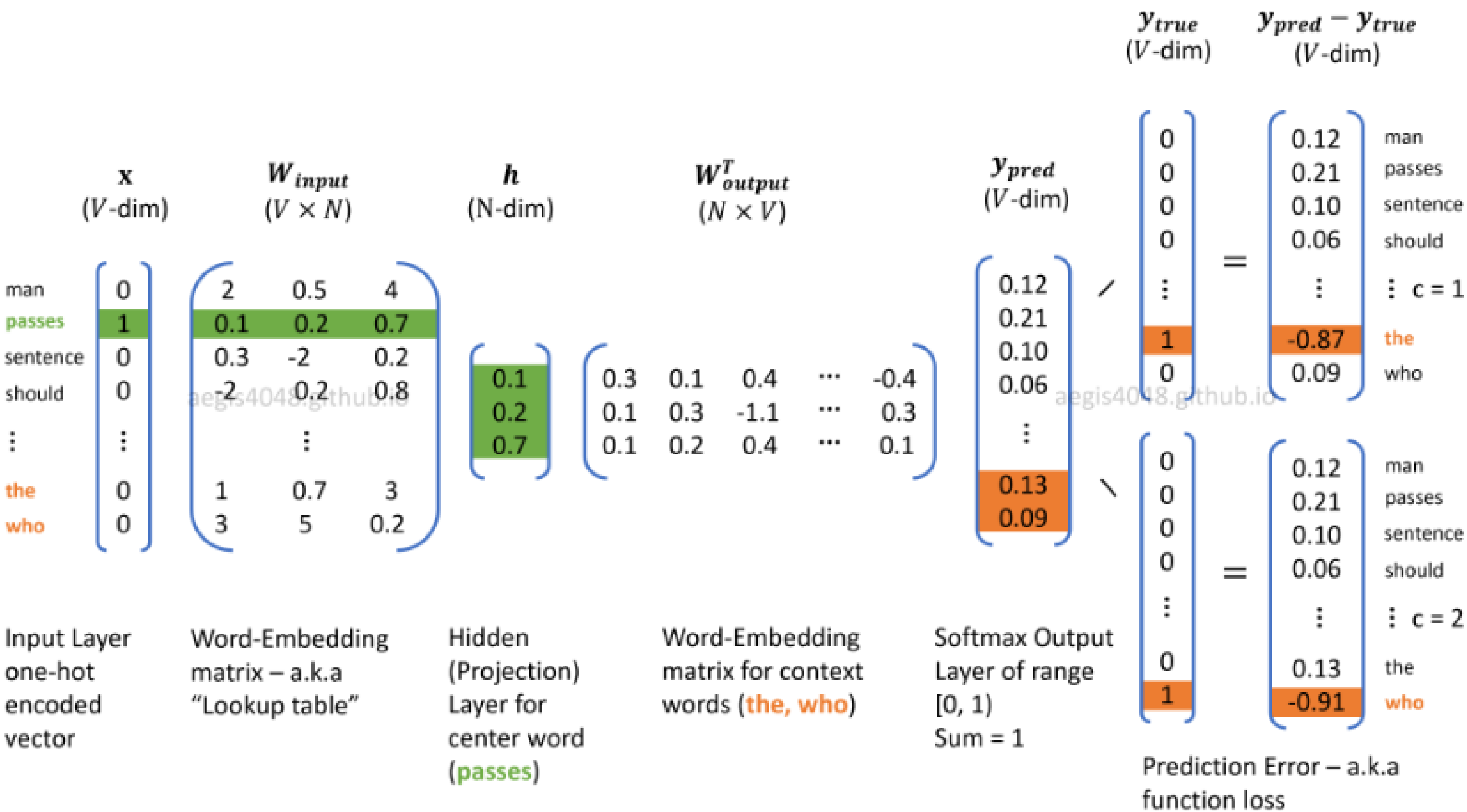
The man who passes the sentence should swing the sword. → (passes, who) (passes, the)

V = 8 (vocabulary dimension)

N = 3 (vector space dimension)

Windows size = 1

Neural Network Structure of Skip-Gram



Source: https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling (Pythonic Excursions)

Neural Network Structure of Skip-Gram

The output of the neural network is a **probability distribution** of dimension *V*, for each of the *V* distinct words of the corpus.

In statistics, the conditional probability of *A* given *B* is indicated by *P (A | B)*

In Skip-Gram, we use the notation *P (Wcontext | Wcenter)* to indicate the conditional probability of observing a word of the context (*Wcontext*), given its central word *(Wcenter)*.

The probability distribution is obtained through the Softmax function

$$\boxed{\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}}$$

$$\begin{bmatrix} p(w_1 | w_{center}) \\ p(w_2 | w_{center}) \\ p(w_3 | w_{center}) \\ \vdots \\ p(w_V | w_{center}) \end{bmatrix} = \frac{exp(W_{output} \cdot h)}{\sum_{i=1}^{V} exp(W_{output_{(i)}} \cdot h)}$$

The Softmax function

Formula: $$\boxed{\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}}$$

The Softmax function compresses a vector of real numbers of size k, in a vector always k-dimensional, of values included in an interval [0,1] whose sum is 1

Example:

- For the vector [*1 ; 2 ; 3 ; 4 ; 1 ; 2 ; 3]*
- the *Softmax* function will be the vector
  [*0,024 ; 0,064 ; 0,175 ; 0,475 ; 0,024 ; 0,064 ; 0 ,175]*

The function assigns most of the weight to the number 4, whose output value is approximately 20 times greater than the value associated at 1

This function therefore highlights the larger values and hides those that are significantly smaller than the maximum value

Word2Vec – Final comments

The model is trained to make predictions, but at the end of the training, the predictive capacity of the network (its output) will be not used, but will be used his internal structure (network weights) to represent the coordinates of each word of the vocabulary in the embedding space

To train and to build a W.E. may be the best approach for a given NLP problem, but it takes a lot of time, a fast computer with lots of RAM and huge disk space and maybe some experience in perfecting input data and training algorithm. An alternative could be to use W.E. models already trained and ready to use; you can find them on the web.

Word2Vec – Final comments

Factors influencing the model performance:

- **size of corpus**: bigger corpora perform better than small ones
- **quality of corpus**: very noisy, fragmented and poorly curated texts generally produce lower quality embedding spaces
- **nature of the corpus**: corpus dealing with specific topics produce W.E. excellent for tasks related to those topics

Queries of W.E. models

«Distance» test

It is a function of questioning the vector space created during the training phase, which returns the ‘n’ closer vectors (words) to a given vector (word searched).

Note: It is also possible to search vectors (words) closest to a vector, that is the linear combination (e.g. sum) of many vectors (words) [it helps in case of polysemic words]

Test *Distance*

```
Enter word or sentence (EXIT to break): roma

Word: roma  Position in vocabulary: 88

                                                   Word       Cosine distance
------------------------------------------------------------------------
                                                 torino              0.719132
                                                palermo              0.642643
                                                 napoli              0.628983
                                                bologna              0.613869
                                          civitavecchia              0.566682
                                                firenze              0.549364
                                                pomezia              0.547425
                                                 milano              0.538685
                                              fiumicino              0.535253
                                                     rm              0.533268
                                             colleferro              0.528270
                                                viterbo              0.524749
                                                catania              0.513795
                                               cagliari              0.506348
                                                trieste              0.504314
                                                bergamo              0.503563
                                               frascati              0.503168
                                              frosinone              0.503045
```

Test *Distance*

```
Enter word or sentence (EXIT to break): roma rieti viterbo

Word: roma  Position in vocabulary: 88

Word: rieti  Position in vocabulary: 4228

Word: viterbo  Position in vocabulary: 5628

                                              Word       Cosine distance
------------------------------------------------------------------------
                                         frosinone              0.672671
                                           perugia              0.644043
                                             terni              0.641548
                                            torino              0.637691
                                            teramo              0.637166
                                            latina              0.622739
                                           cosenza              0.616780
                                           bologna              0.616200
                                           catania              0.609594
                                           palermo              0.600828
                                        colleferro              0.597391
                                           firenze              0.594622
                                           bergamo              0.591040
                                           pescara              0.588208
                                           aprilia              0.581499
```

«Word-Analogy» test

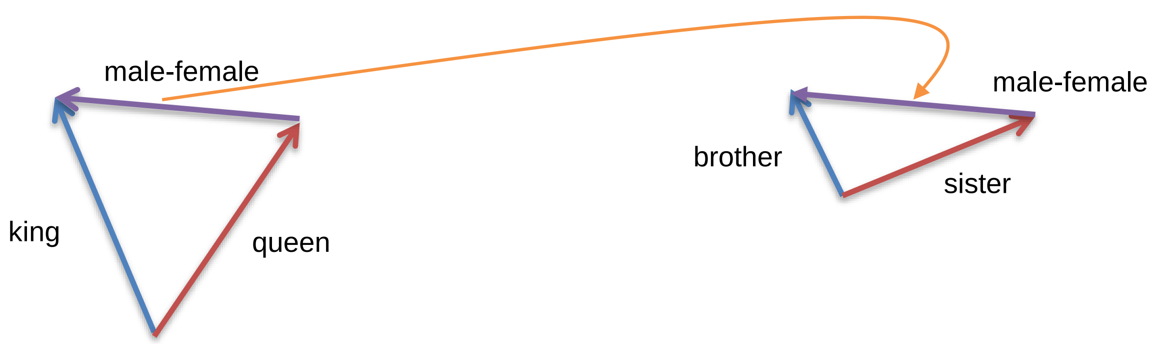
It is a function of questioning the vector space that, with the insertion of three vectors (words), returns the most representative words to solve the analogy.

e.g.

king *: queen = male : female*

where the relation 'male-female' applied to the word 'queen' to obtain the word 'king' [king = queen + (male - female)],

it can be applied to other words



Word Analogy test

List of some natural language relationships 'captured' by the W.E. model

- Syntactic
  - ✓ male - female
  - ✓ singular - plural
  - ✓ superlatives - diminutive
  - ✓ synonyms – opposite

- Semantics:
  - ✓ man - woman
  - ✓ nationatility - nation
  - ✓ nation – capital city
  - ✓ region – regional capital city
  - ✓ initial province - province
  - ✓ political - party of belonging
  - ✓ newspaper – city of reference
  - ✓ …………
  - ✓ …………

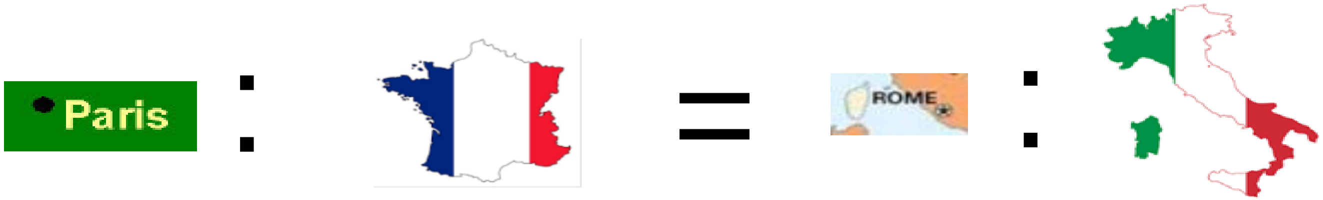
«Accuracy Top 1» test

It's a *Word Analogy test* that tries to solve a list of syntactic and semantic analogy, organized in groups, and reported into a benchmark file.

The test is passed only if the correct word is the first of those proposed to complete the analogy

Word Embedding potentiality

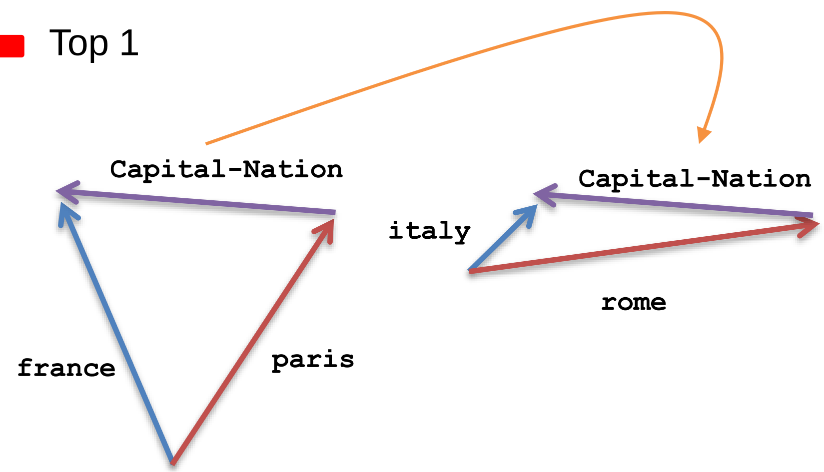
Analogy Test – relationship: Capital - Nation




```
bash-4.2$  ./word-analogy  models/GoogleNews-vectors-negative300.bin
Enter three words (EXIT to break): paris  france  rome
```

```
        Word                   Distance
------------------------------------------
       italy                   0.519953
    european                   0.507585
     italian                   0.505775
         epl                   0.490745
       spain                   0.488867
     england                   0.485267
    italians                   0.484242
      kosovo                   0.481350
     lampard                   0.480774
       malta                   0.478857
        juve                   0.478361
   newcastle                   0.477507
     glasgow                   0.477125
      europe                   0.474660
```

Top 1



Word Embedding potentiality

Analogy Test – relationship: *Singular - Plural*

 :  =  : 

```
bash-4.2$  ./word-analogy  models/GoogleNews-vectors-negative300.bin
Enter three words (EXIT to break): boy  boys  car
```

Word	Distance
cars	0.669246
vehicle	0.562532
vehicles	0.524835
Car	0.523344
Subaru_WRX	0.509807
Ford_Focus	0.503423
Porsches	0.496703
Honda_Civic	0.488979
SUV	0.485614
sedan	0.481740
Chevrolet	0.480740

Top 1

Word Embedding potentiality

Analogy Test – relationship: Nation - Newspaper


:


=

:



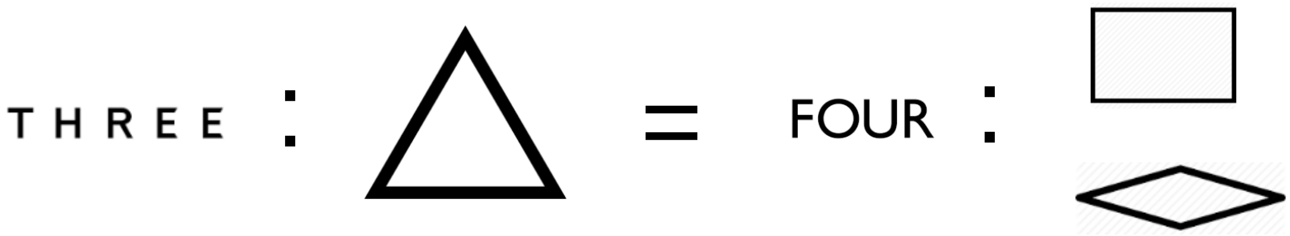
```
bash-4.2$  ./word-analogy  models/GoogleNews-vectors-negative300.bin
Enter three words (EXIT to break): Germany  Der_Spiegel  England

                                              Word              Distance
------------------------------------------------------------------------
                                   Daily_Telegraph              0.521107
                                        Daily_Mail              0.506416
                                          Guardian              0.490951
                                    Wisden_Cricket              0.485638
                                  ticker_symbol_BNK             0.483723
                                  Wisden_Cricketer              0.482764
                                   stock_symbol_BNK             0.481484
                              Gloucestershire_Echo              0.475339
                                       Scyld_Berry              0.468583
                      Newcastle_Evening_Chronicle               0.457495
                                      LSO_St_Lukes              0.456623
                                         Telegraph              0.456241
                           London_Evening_Standard              0.456229
                                       Mike_Selvey              0.444308
```

Top 1

Word Embedding potentiality

Analogy Test – relationship: Geometric Figures




```
bash-4.2$  ./word-analogy  models/GoogleNews-vectors-negative300.bin
Enter three words (EXIT to break): three  triangle  four

                  Word                Distance
---------------------------------------------
             triangles                0.524841
  equilateral_triangle                0.516976
             rectangle                0.515894
               rhombus                0.514743
           equilateral                0.506330
    isosceles_triangle                0.478837
            concentric                0.468067
           semicircles                0.457456
               hexagon                0.453780
```

Top 4

GloVe

Global Vectors for Word Representation

GloVe

It’s one of the models used for the production of Word Embeddings

It combines the advantages of the two major families of NLP models in the literature:

- *global matrix factorization (latent semantyc analysys - LSA)*
- *local context window methods (Word2Vec)*

The first family of models is based on the c.d. term-document-matrix; they are able to exploit the statistical information of the texts (counts, word frequency distribution, etc.), but they are not suitable for solving analogies

The second family of models is very efficient in solving analogies, but, before the advent of GloVe, it didn’t use statistical information (in explicit manner)

GloVe

GloVe = Global Vector means that in producing the model we have to consider:

- in addition to the foundation of Word2Vec for which close words from a syntactic / semantic point of view, co-occur in the same context
- also the statistics on the co-occurrence at global level in the corpus

The co-occurrence statistics show a primary source of information and unlike Word2Vec, GloVe exploits them explicitly, implementing in his algorithms the ***matrix of word-word co-occurrence***

$\boldsymbol{X} \quad = \textit{matrix of word-word co-occurence counting}$

$\boldsymbol{X_{ij}} = \textit{number of times that word } j \textit{ appear in the context of word } i$

$\boldsymbol{X_i} = \sum_{K} \boldsymbol{X_{ik}}$ *Sum of frequencies of the k words in the i context*

X_{ij}		j						
		ice	steam	solid	gas	water	fashion	
i	ice	0	3	10	1	24	2	**40**
	steam	2	0	2	12	15	0	**31**
	solid	20	2	0	3	3	0	**28**
	gas	2	12	0	0	3	0	**17**
	water	20	14	2	3	0	1	**40**
	fashion	2	1	1	1	1	0	**6**

Co-occurence matrix potentiality (1/2)

The main intuition underlying the GloVe model is the simple observation that **ratios of word-word co-occurrence probabilities** have the potential for encoding some form of meaning

$$P_{ij} = P(j/i) = \frac{X_{ij}}{X_i}$$

Probability that the word j appear in the context of the word i

i.e.: P(ice | water) = 24 / 40

P_{ij}		j						
		ice	steam	solid	gas	water	fashion	
i	ice	-	3 / 40	10 / 40	1 / 40	24 / 40	2 / 40	1
	steam	2 / 31	-	2 / 31	12 / 31	15 / 31	-	1
	solid	20 / 28	2 / 28	-	3 / 28	3 / 28	-	1
	gas	2 / 17	12 / 17	-	-	3 / 17	-	1
	water	20 / 40	14 / 40	2 / 40	3 / 40	-	1 / 40	1
	fashion	2 / 6	1 / 6	1 / 6	1 / 6	1 / 6	-	1

Co-occurence matrix potentiality (2/2)

Suppose we want to analyze the relation between the two words *i = ice and j = steam*; we can study this relation analyzing the ratio between their probability to co-occour with the probe word **k** ➔ P_{ik} / P_{jk}

where

$P_{ik} = P(k/i) = X_{ik} / X_i$ *probability that word **k** appear in the context of word **i***

$P_{jk} = P(k/j) = X_{jk} / X_j$ *probability that word **k** appear in the context of word **j***

- for probe words related to *i* but not to *j* P_{ik} / P_{jk} will be large (es.: *solid*)
- for probe words related to *j* but not to *i* P_{ik} / P_{jk} will be small (es.: *gas*)
- for probe words that are either related to both i and j or neither P_{ik} / P_{jk} will be close to 1 (es.: *water, fashion*)

P_{ik} / P_{jk}	**probe words (k)**		
	solid	**gas**	**water**
i = ice **j = steam**	(10/40) / (2/31) = 3,75	(1/40) / (12/31) = 0,06	(24/40) / (15/31) = 1,24

Global Vectors for Word Representation - GloVe

GloVe has been designed to make explicit what Word2Vec did implicitly. It directly exploits words co-occurrences to define the structure of the embedding space:

- High co-occurrency words are mapped to almost parallel vectors
- Low co-occurrency words are mapped to almost perpendicular vectors

Indeed, GloVe minimizes the following loss function:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

where:

- $w_i^T \tilde{w}_j$ → is the dot product of the image vectors of words *i* and *j*

The dot product has a maximum for parallel words and a minimum for orthogonal ones.

- X_{ij} → is the *i*, *j* element of the ***matrix of word-word co-occurrence***

i, *j* span over the whole vocabulary of the corpus, but the co-occurrences are always calculated using a sliding window that defines the context. The matrix is calculated only once at the beginning, and then remains fixed during the training.

FastText

Fast Text

- It was developed by Facebook Research in 2016
- in addition to the words it also uses a lower level, its characters or n-grams, as input for creating W.E. (in some ways, a word becomes the context of itself)
- it introduce the concept of hierarchy (among characters and words): a word represents a label for its n-grams
- it can create W.E. models, starting from smaller data sets (small corpus)
- the training phase is much faster than Word2Vec and GloVe
- it implements the generalization. It means that the models generated by FastText are able to represent in the embedding space the words not included in the starting corpus, too

Word2Vec, GloVe and FastText in one sentence

- ***Word2Vec***: is founded on the concept that **similar words appear in similar contexts**

- ***GloVe*** extends Word2Vec introducing **statistics** on the words' co-occurrences in the corpus, at **global level**

- ***FastText*** improves Word2Vec **generalizing** the model to unknown words, that are not included in the input corpus

W.E. experiences in Istat

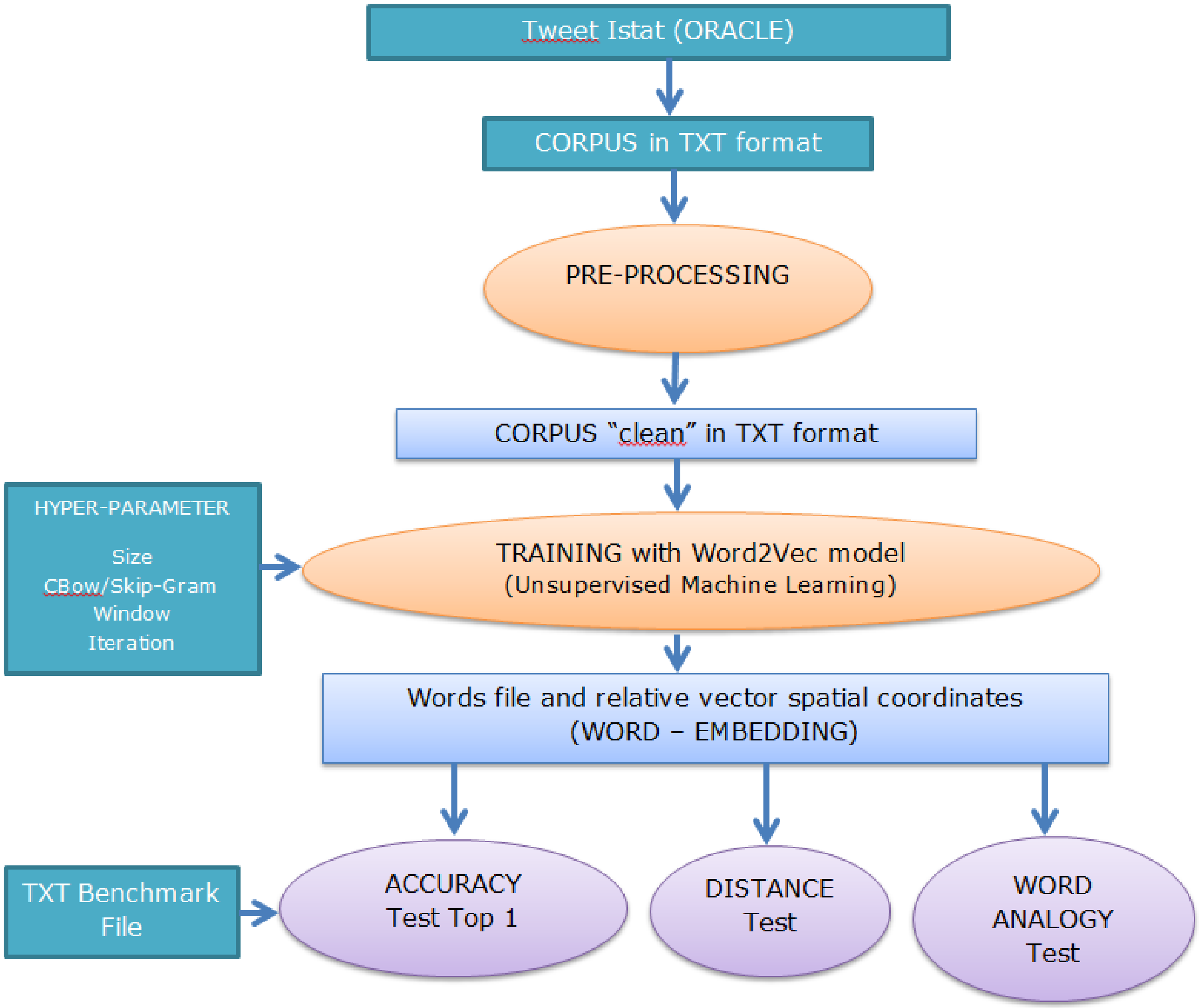
Why Word Embedding in Istat ?

- Eurostat encourages the experimentation of new techniques that use Big Data for the production of official statistics (Scheveningen Memorandum, February 2014)

- Among the Big Data sources, textual ones are certainly among the most available (e.g. text coming from Social Media or web scraping activities)

- One of the latest techniques in computational linguistics subject was Word Embeddings

Flow chart of processing activities of Italian Twitter



Pre-Processing

It consists of the following "cleaning“ activities of the corpus, before it is passed to the next stage of training.

Remove of:

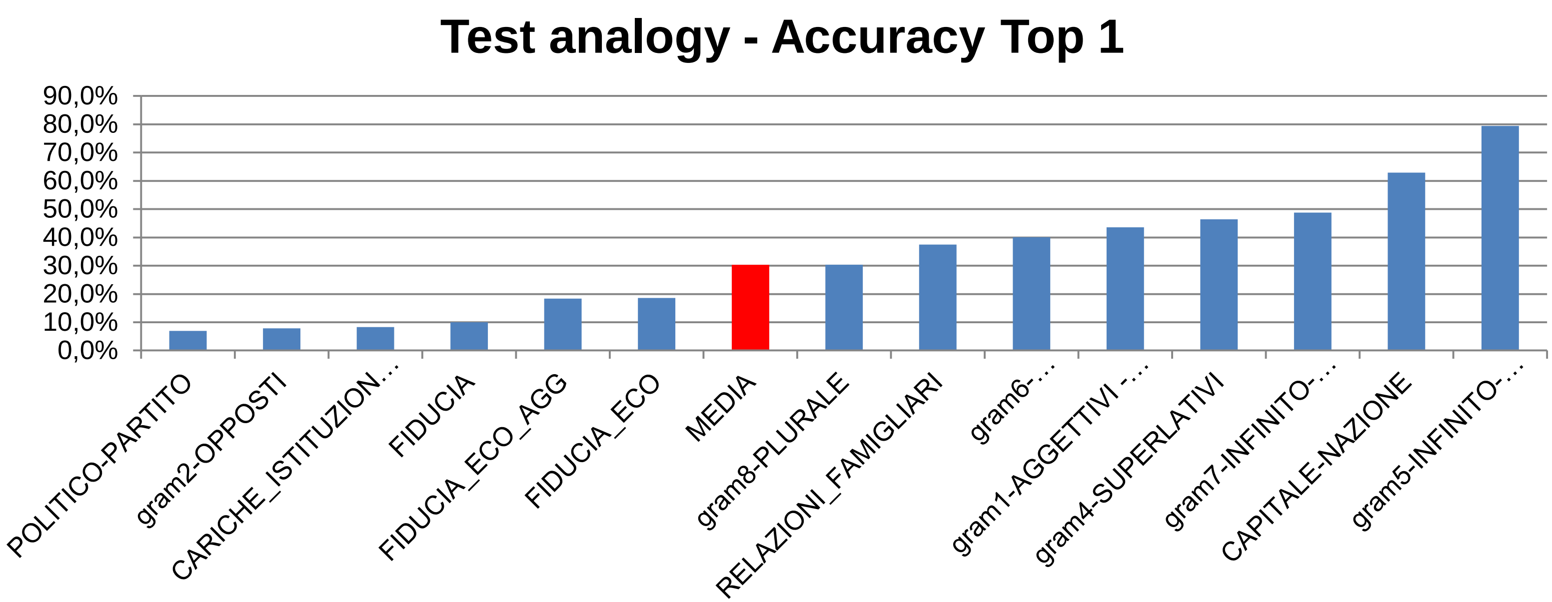
- *URL*
- *hashtag*
- *special characters*
- *Double spaces*
- *Word of one character*

Features of “our” word embedding model

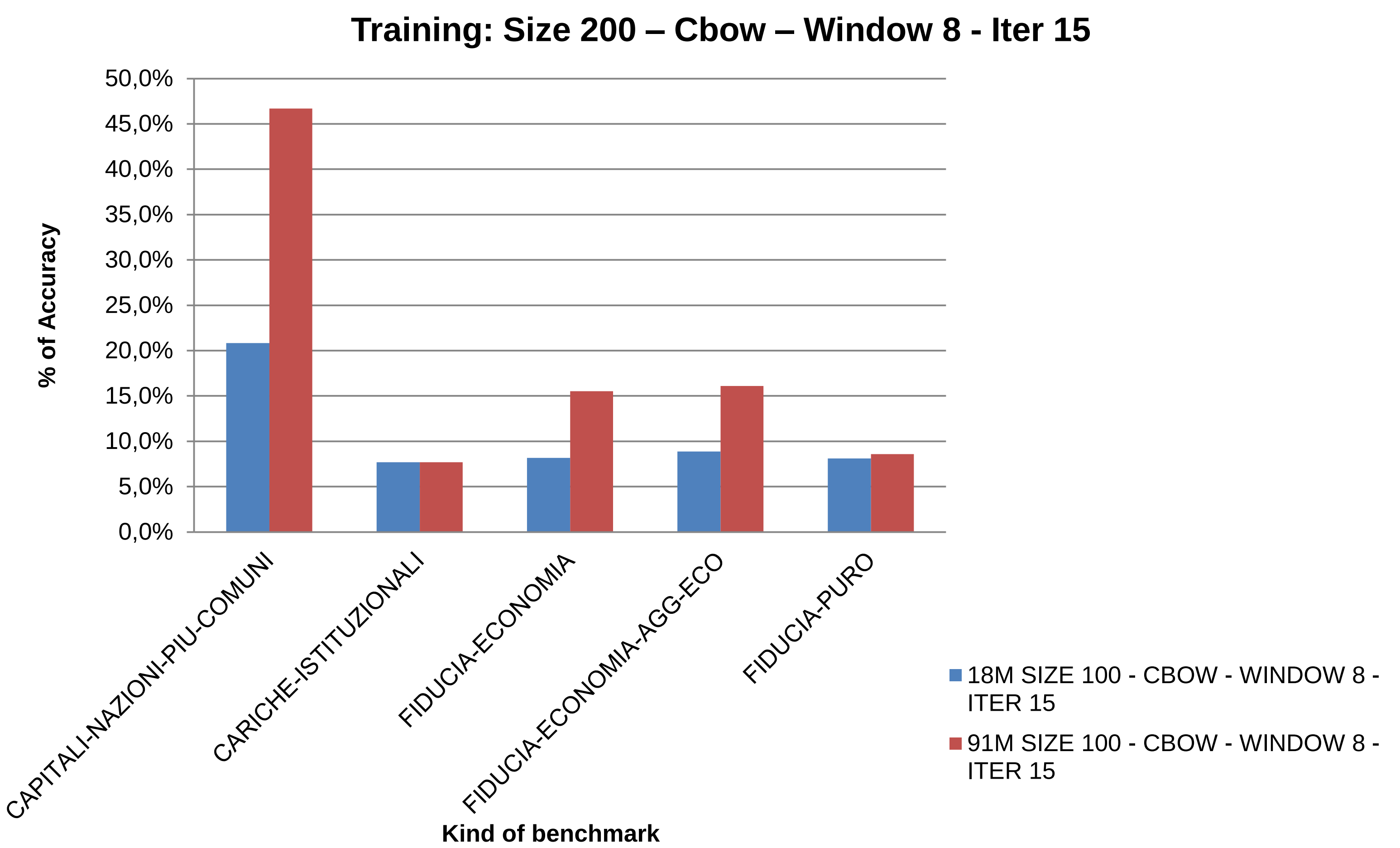
Dimension: 140 millions of tweets (period: February 2016 – June 2018)

Type of corpus: tweets

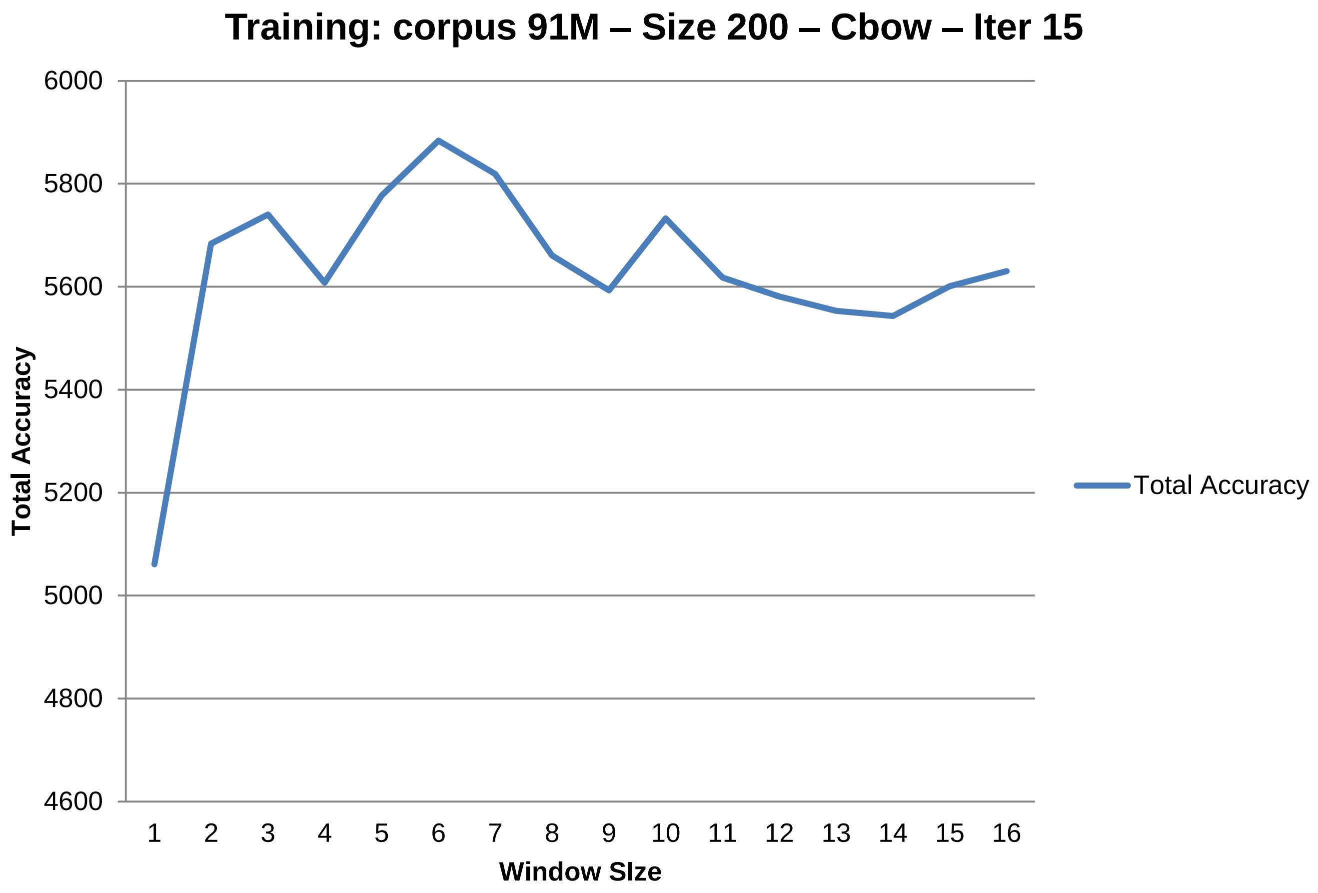
- deal on sport, current events and politics topics
- have a maximum length of 140/280 characters (from September 2017)



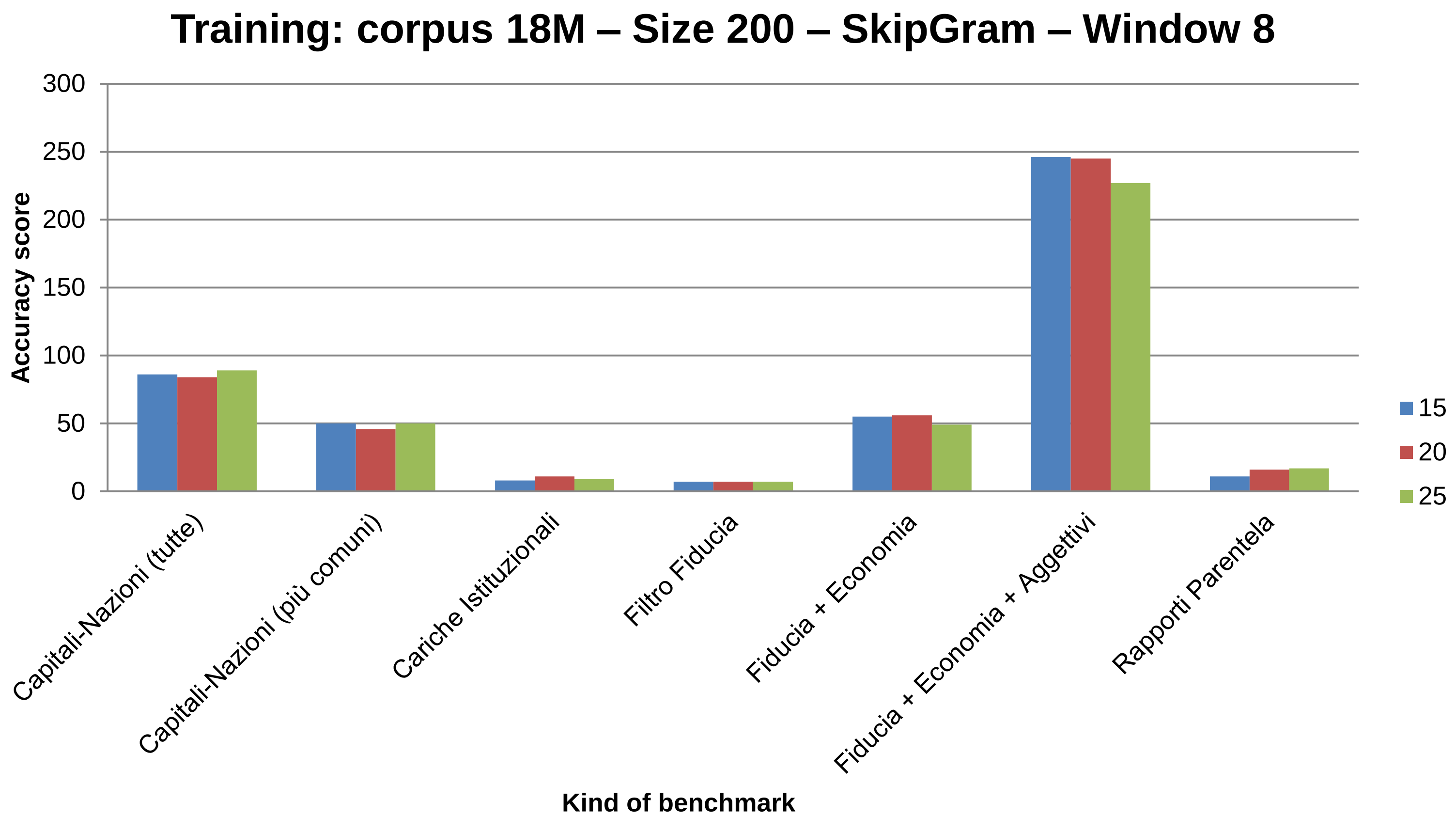
Accuracy test : corpus dimension



Accuracy test : window size



Accuracy test : number of iteration



Reached results

- We have tested Word Embedding techniques based on the Word2Vec algorithm on a large *corpus* of textual data in Italian language: the collection of ~ 90 million Tweets caught in about a year by our data collection procedures from Twitter

- We studied the quality of the results obtained by varying the main hyper-parameters of the Word2Vec algorithm by selecting the set that provides maximum *accuracy top1*

- We built the Word Embedding models on Tweet captured both the 'Trusted Filter' and 'Istat Filter' using the best hyperparameters

Representation of W.E. models

Exploring and visualizing big embedding models through graphs:

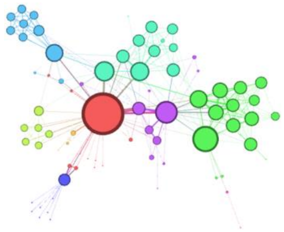
Targets

- to explore the model around a semantic area and to represent the relationships between words emerging from specific corpus

Issues

- The WE model has a high dimension in the order of 200/300 vector coordinates
- There are millions of words, therefore millions of vectors in the embedding space

We need a tool that help us to perform this task 

Use Graphs 

The graphs allow

- a visualization of the model in two dimensions
- a representation of relationships between entities (words)

Why graphs ?

Word embeddings are vectors represented in a very high dimensional space.

How we can visualize and handle these objects ?

Statistical methods to reduce space dimensionality maintaining the relationships between vectors:

- PCA
- T-SNE

Our proposal is to introduce the expressive power of graphs to represent word embedding.

What is a graph?

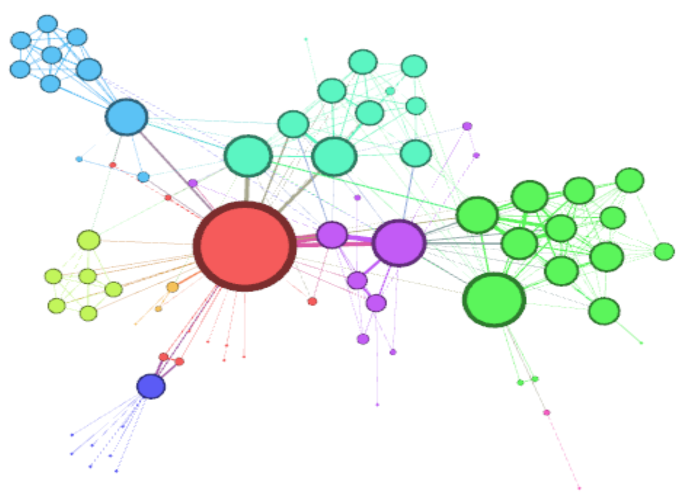
A graph is a mathematical structure consisting of a set of nodes and links between them

Nodes represent entities

The edges (also called links or lines) represent relationships between entities

DIRECTED

UNDIRECTED



A word embedding graph

For a WE model:

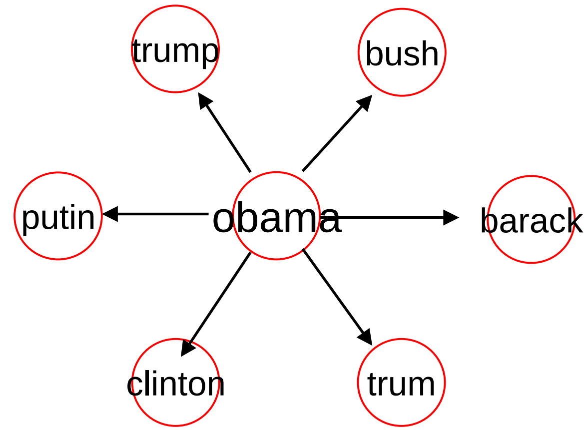
- **words** are **nodes**
- **distance relationships** are **edges** (also called links or lines)

Each word is connected to the first W words nearest in the cosine distance metric

```
model.wv.most_similar(positive=['obama'],topn=100)
```

```
[(u'trump', 0.7187066078186035),
 (u'bush', 0.6275098323822021),
 (u'barack', 0.6016936302185059),
 (u'trum', 0.595779538154602),
 (u'clinton', 0.5875608921051025),
 (u'putin', 0.579934298992157),
 (u'hillary', 0.5499014854431152),
 (u'incredibilevedere', 0.5437788367271423),
 (u'obam', 0.5359461307525635),
 (u'donald', 0.5123184323310852),
 (u'washington', 0.5050041675567627),
 (u'hollande', 0.49855348467826843),
 (u'russia', 0.4958000183105469),
```

W width is a parameter that indicates the number of connected words



GEOMETRIC graph

Parameters:

SEEDS = set of words from which the exploration starts

W = number of closest words to include at each iteration

N = Iteration number

If exploration always finds new words (i.e. there are no cycles), the number of nodes grows exponentially with the iterations. N_2=1+3^1+3^2

NOTE:
The semantic relationships between the SEEDS and the words included at iteration *k* decreases quickly with *k*

W=3

N=2

GEOMETRIC graph

Parameters:

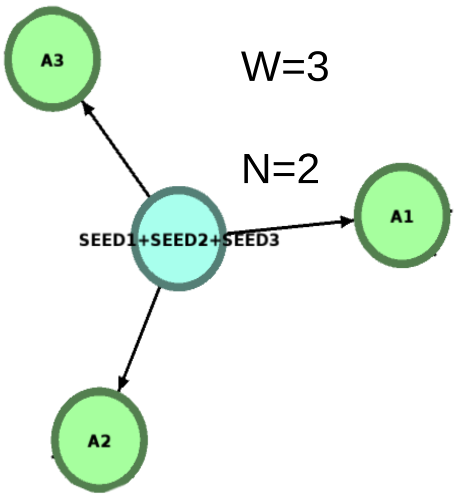
SEEDS = set of words from which the exploration starts

W = number of closest words to include at each iteration

N = Iteration number

If exploration always finds new words (i.e. there are no cycles), the number of nodes grows exponentially with the iterations. N_2=1+3^1+3^2

NOTE:
The semantic relationships between the SEEDS and the words included at iteration *k* decreases quickly with *k*



GEOMETRIC graph

Parameters:

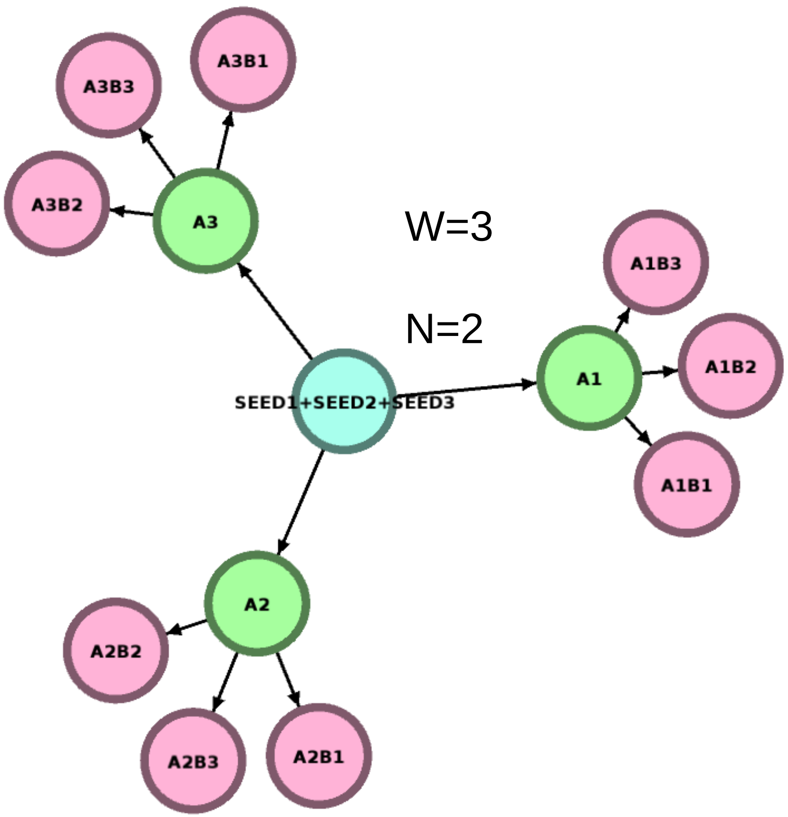
SEEDS = set of words from which the exploration starts

W = number of closest words to include at each iteration

N = Iteration number

If exploration always finds new words (i.e. there are no cycles), the number of nodes grows exponentially with the iterations. N_2=1+3^1+3^2

NOTE:
The semantic relationships between the SEEDS and the words included at iteration k decreases quickly with k



LINEAR graph

Parameters:

SEEDS = set of words from which the exploration starts

W = number of closest words to include at each iteration

N = Iteration number

At each iteration, a virtual node is created consisting of the sum of the words found.

The number of nodes, grows linearly with the iterations

N_i =1+ W * i

NOTE:
The graph is easy to read and tends to identify a “semantic direction” within the embedding space, as if a story is being told

W=2

N=2

LINEAR graph

Parameters:

SEEDS = set of words from which the exploration starts

W = number of closest words to include at each iteration

N = Iteration number

At each iteration, a virtual node is created consisting of the sum of the words found.

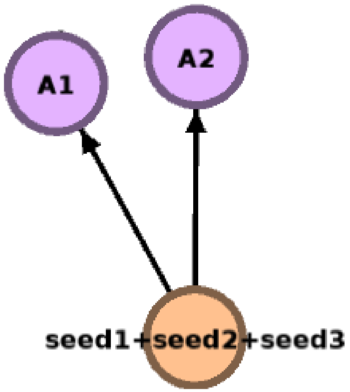
The number of nodes, grows linearly with the iterations

N_i =1+ W * i

NOTE:
The graph is easy to read and tends to identify a “semantic direction” within the embedding space, as if a story is being told

W=2

N=2



LINEAR graph

Parameters:

SEEDS = set of words from which the exploration starts

W = number of closest words to include at each iteration

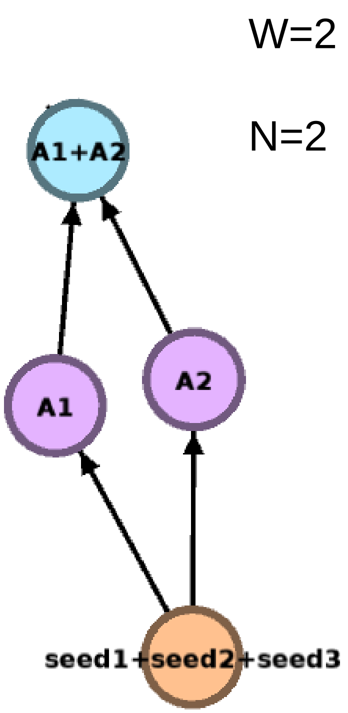
N = Iteration number

At each iteration, a virtual node is created consisting of the sum of the words found.

The number of nodes, grows linearly with the iterations

N_i =1+ W * i

NOTE:
The graph is easy to read and tends to identify a “semantic direction” within the embedding space, as if a story is being told



LINEAR graph

Parameters:

SEEDS = set of words from which the exploration starts

W = number of closest words to include at each iteration

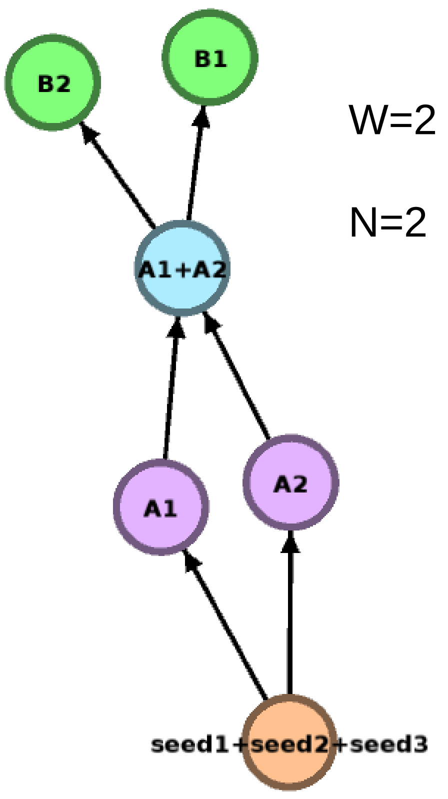
N = Iteration number

At each iteration, a virtual node is created consisting of the sum of the words found.

The number of nodes, grows linearly with the iterations

N_i =1+ W * i

NOTE:
The graph is easy to read and tends to identify a "semantic direction" within the embedding space, as if a story is being told



GEOMETRIC ORIENTED graph

Parameters:

SEEDS = set of words from which the exploration starts

W = number of closest words to include at each iteration

N = Iteration number

At each iteration, each found word generates a virtual node which is the sum of all the words belonging to non-virtual nodes along the shortest path connecting the current word to the SEEDS node

The graph is compact and full of cyclic paths

Exploration remains in the initial semantic area

W=2

N=2

GEOMETRIC ORIENTED graph

Parameters:

SEEDS = set of words from which the exploration starts

W = number of closest words to include at each iteration

N = Iteration number

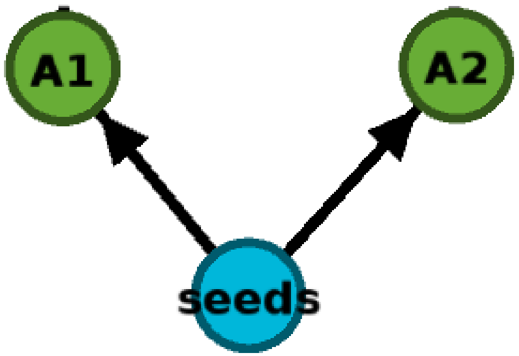
At each iteration, each found word generates a virtual node which is the sum of all the words belonging to non-virtual nodes along the shortest path connecting the current word to the SEEDS node

The graph is compact and full of cyclic paths

Exploration remains in the initial semantic area

W=2

N=2



GEOMETRIC ORIENTED graph

Parameters:

SEEDS = set of words from which the exploration starts

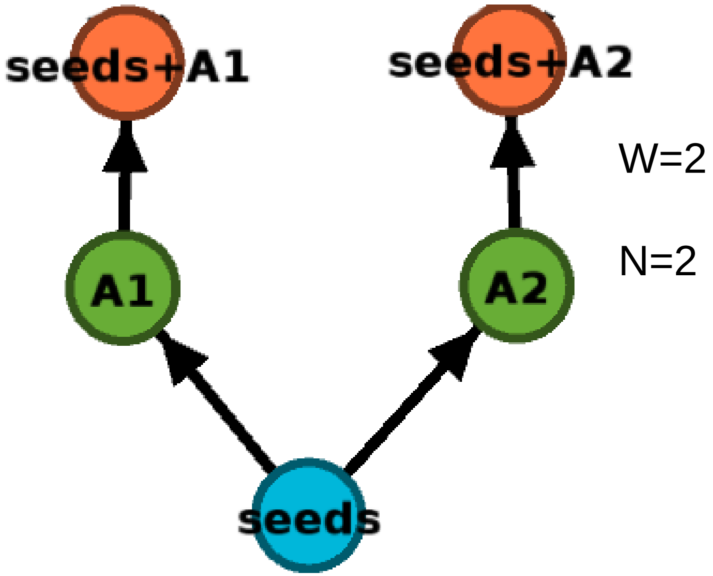
W = number of closest words to include at each iteration

N = Iteration number

At each iteration, each found word generates a virtual node which is the sum of all the words belonging to non-virtual nodes along the shortest path connecting the current word to the SEEDS node

The graph is compact and full of cyclic paths

Exploration remains in the initial semantic area



GEOMETRIC ORIENTED graph

Parameters:

SEEDS = set of words from which the exploration starts

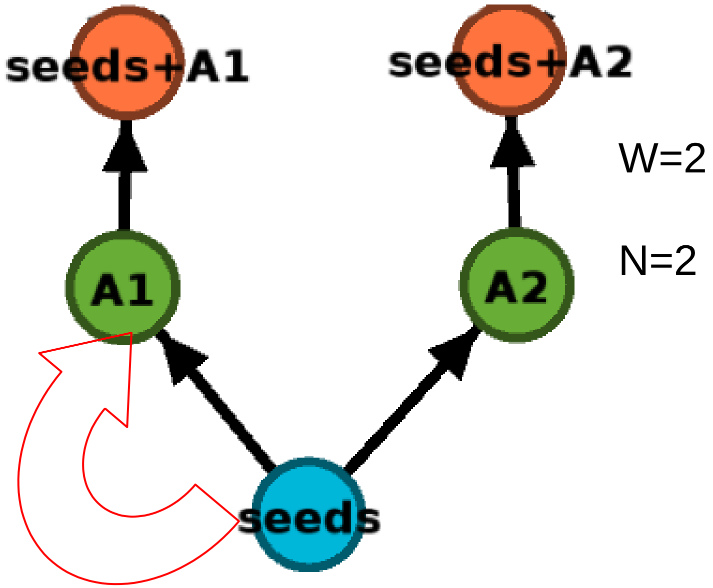
W = number of closest words to include at each iteration

N = Iteration number

At each iteration, each found word generates a virtual node which is the sum of all the words belonging to non-virtual nodes along the shortest path connecting the current word to the SEEDS node

The graph is compact and full of cyclic paths

Exploration remains in the initial semantic area



GEOMETRIC ORIENTED graph

Parameters:

SEEDS = set of words from which the exploration starts

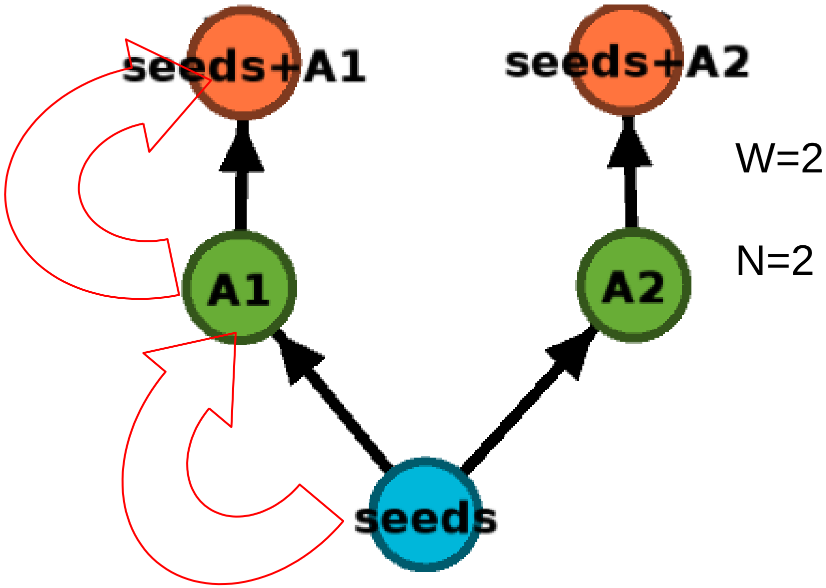
W = number of closest words to include at each iteration

N = Iteration number

At each iteration, each found word generates a virtual node which is the sum of all the words belonging to non-virtual nodes along the shortest path connecting the current word to the SEEDS node

The graph is compact and full of cyclic paths

Exploration remains in the initial semantic area



GEOMETRIC ORIENTED graph

Parameters:

SEEDS = set of words from which the exploration starts

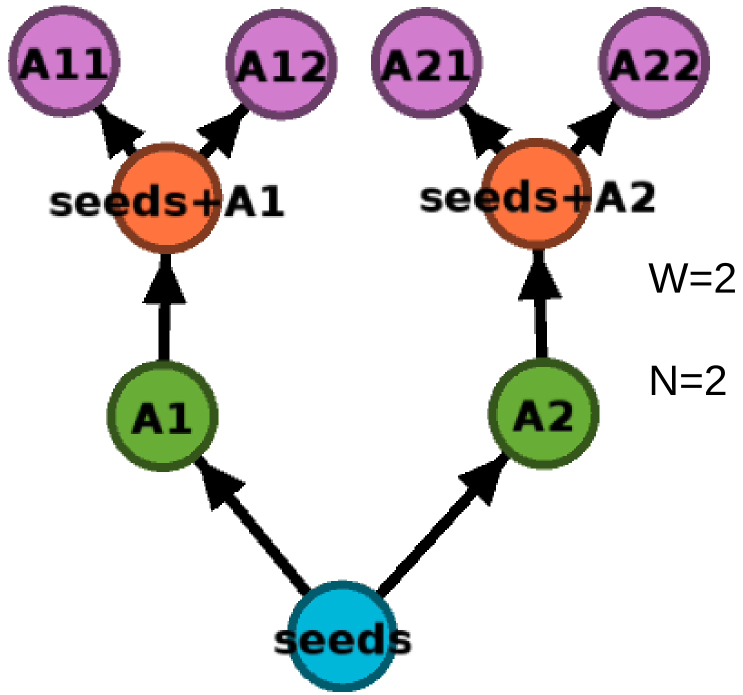
W = number of closest words to include at each iteration

N = Iteration number

At each iteration, each found word generates a virtual node which is the sum of all the words belonging to non-virtual nodes along the shortest path connecting the current word to the SEEDS node

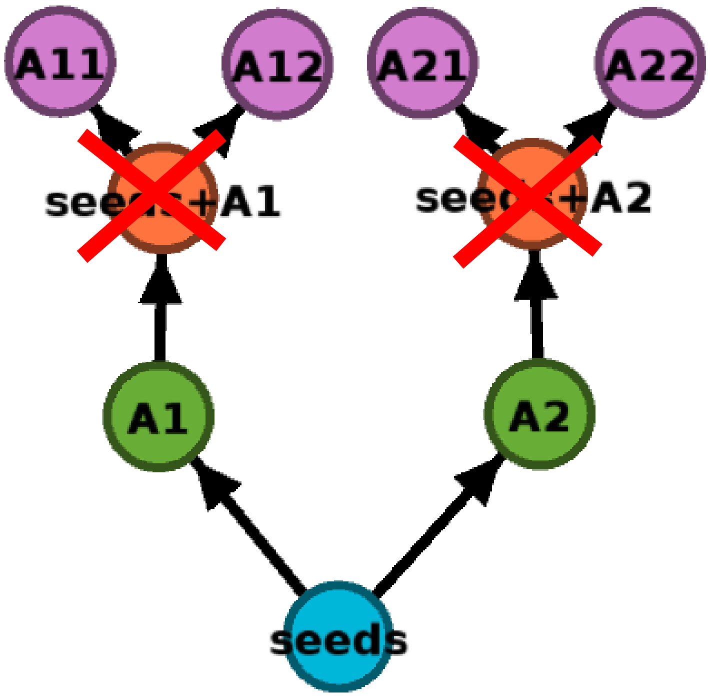
The graph is compact and full of cyclic paths

Exploration remains in the initial semantic area



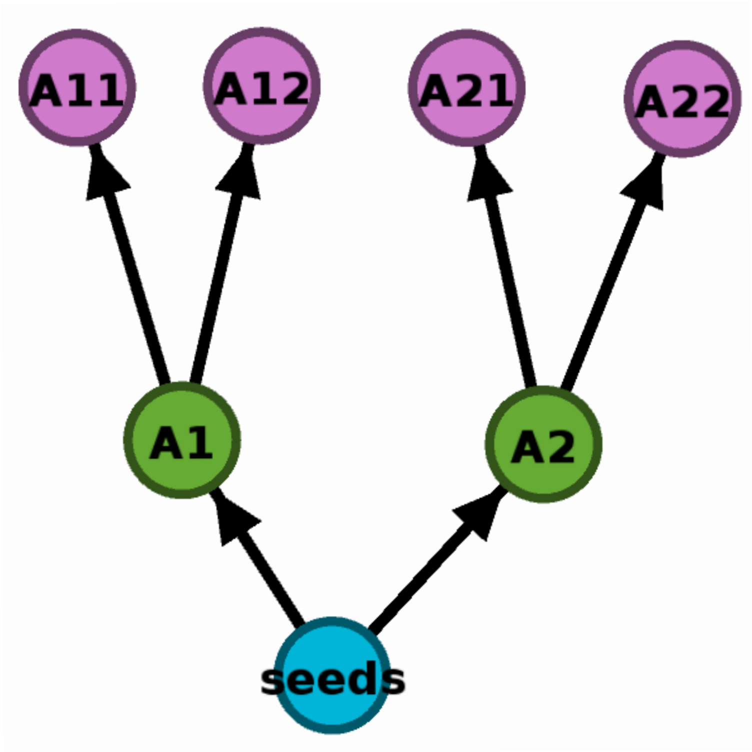
GEOMETRIC ORIENTED graph

Finally, to make the graph more readable, the virtual nodes are eliminated.



GEOMETRIC ORIENTED graph

Finally, to make the graph more readable, the virtual nodes are eliminated.



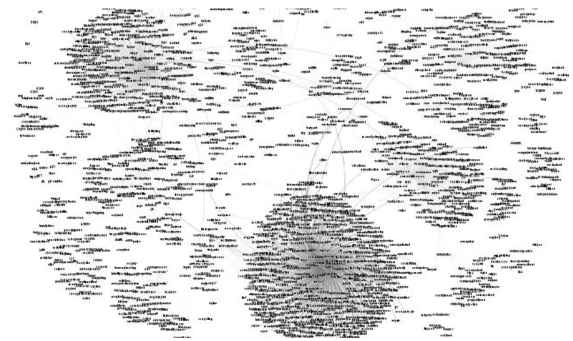
Applications

Use case 1: select the best key words used to filter the Twitter API; the aim is to realize a new filter on ‘Europe’ topic.

Big Data Source

Twitter collection

WORD EMBEDDING MODEL

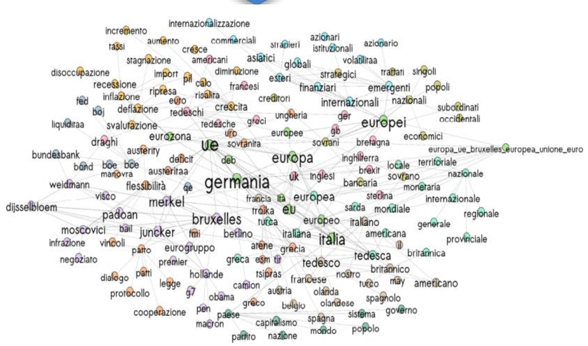


seed words = [europa + ue + unione + europea + bruxelles + euro]

Other parameters:

- kind of Graph (Geometric, Geo-Oriented, Linear)
- number of closest words (W)
- iteration number (N)

Graph Generation



‘Filter’ Words

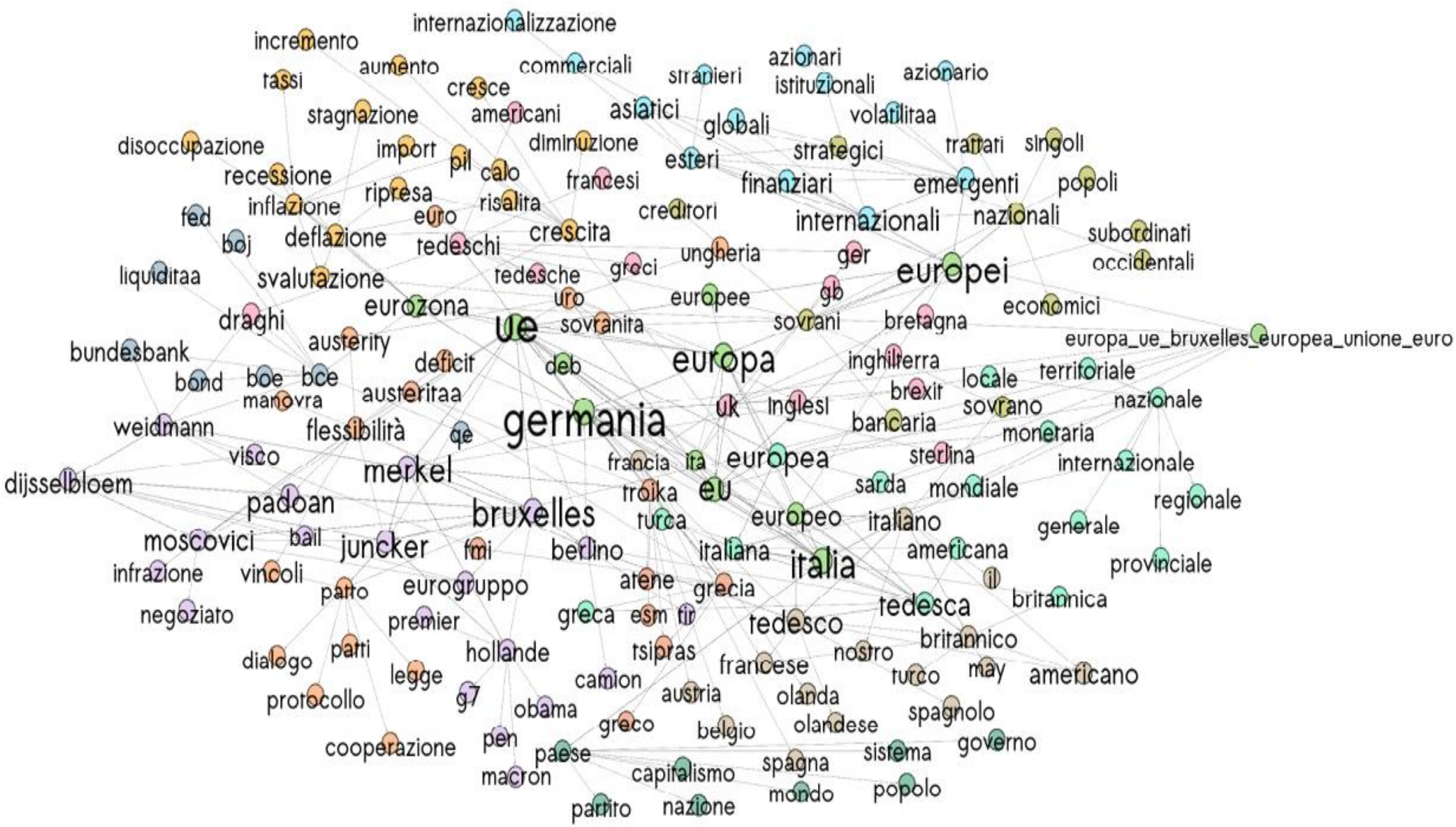
germania
europa
europei
ue
bruxelles
italia
merkel
sovranità
juncker
europee
sovrano
austerità
eurozona

Use case 1: Geometric Graph exploration

seed words = [europa + ue + unione + europea + bruxelles + euro]

W (number of closest words to include at each iteration) = 8

N (iteration number) = 3

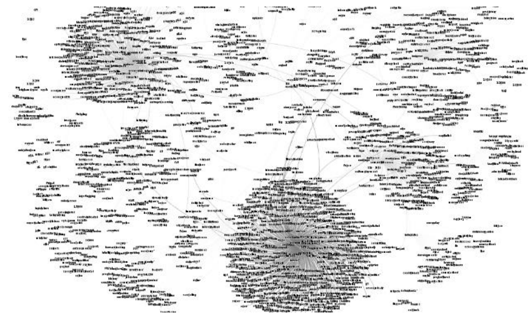


Use case 2: to use a word embedding model as an efficient word encoding layer to form an input for a Deep-Learning Classifier to classify enterprise web sites in e-commerce or not (1/2)

E-Commerce

Web sites

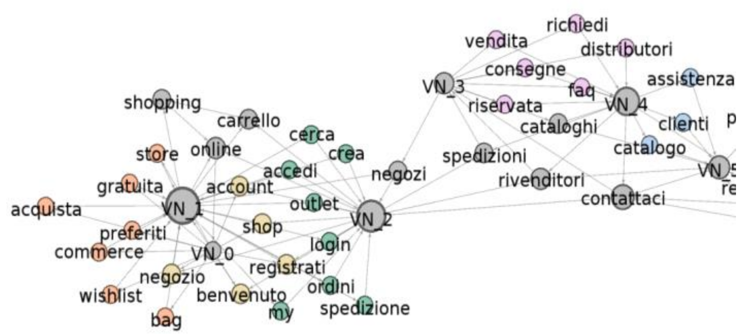
WORD EMBEDDING MODEL



seed words = =
[shopping + online + carrello]

Other parameters:
- kind of Graph (Geometric, Geo-Oriented, Linear)
- number of closest words (W)
- iteration number (N)

Graph Generation



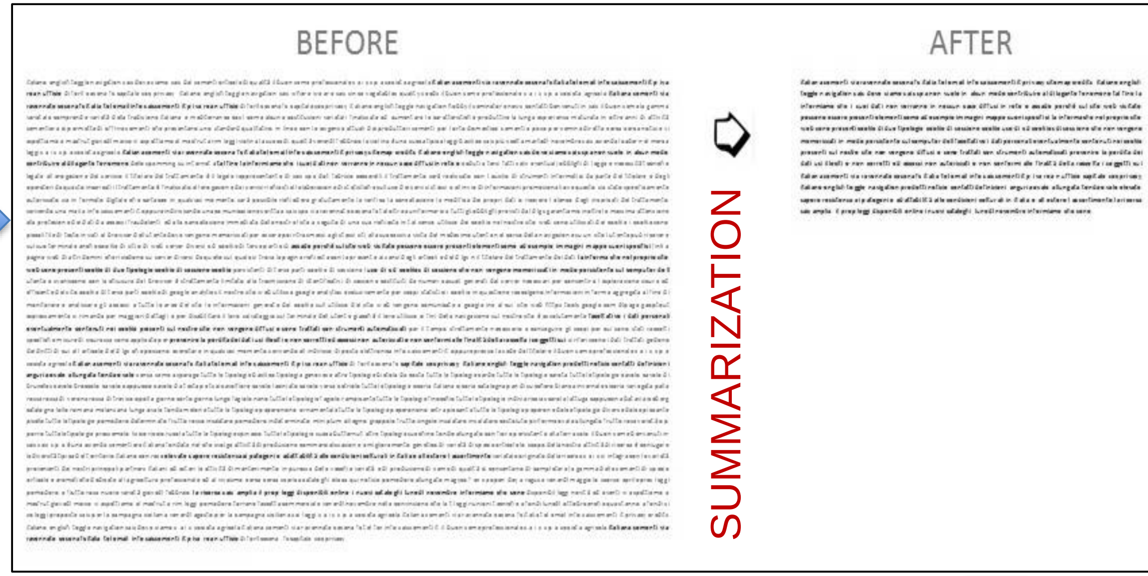
Ontology 'E-Commerce'

shopping
online
negozi
carrello
preferiti
acquista
outlet
ordini
wishlist
spedizione
vendita
store
………
……..

Use case 2: to use a word embedding model as an efficient word encoding layer to form an input for a Deep-Learning Classifier to classify enterprise web sites in e-commerce or not (2/2)

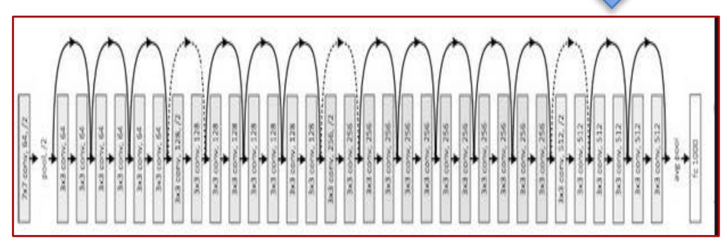
Ontology 'E-Commerce'

shopping
online
negozi
carrello
preferiti
acquista
outlet
ordini
wishlist
spedizione
vendita
store
....
....



1 - E-Commerce

1 – NON E-Commerce



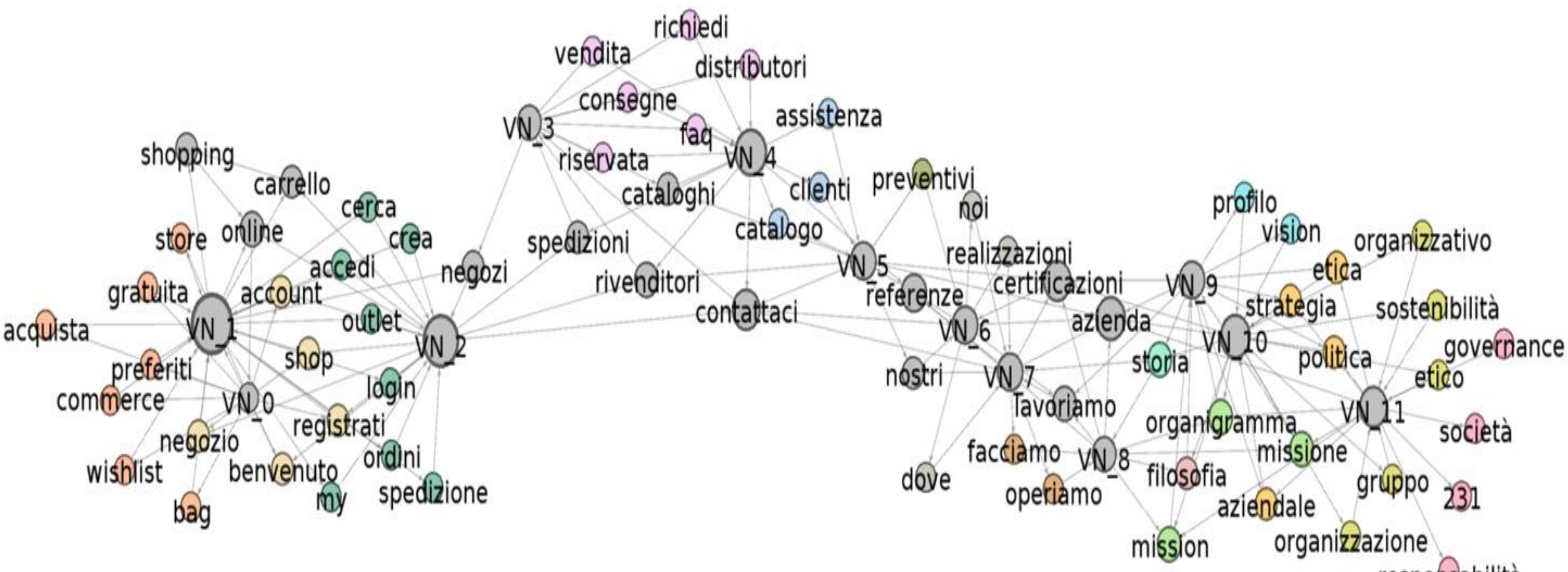
Deep Learning Classifier

Use case 2: Linear Graph exploration

seed words = [shopping+ online + carrello]

W (number of closest words to include at each iteration) = 12

N (iteration number) = 11



References

«Proceedings of the 14th international conference on statistical analysis of textual data (pagg. 174-182) - “Word Embeddings: a Powerful Tool for Innovative Statistics at Istat” (ISBN: 978-88-3293-137-2) – Authors: Fabrizio De Fausti - Massimo De Cubellis – Diego Zardetto

“Efficient Estimation of Word Representations in Vector Space” – Authors: Tomas Mikolov, Kai Chen, Greg Corrado,Jeffrey Dean

https://nlp.stanford.edu/projects/glove/

https://www.aclweb.org/anthology/N13-1090.pdf

https://github.com/tmikolov/word2vec

WORD EMBEDDING MODEL