



UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea Magistrale in Informatica

Tecnologie per il Web Semantico

Embedding di entità in un grafo RDF

Prof. Fanizzi Nicola

Christopher Piemonte

659723

a.a. 2016/2017

Indice

Indice	1
1 - Il Web Semantico	2
2 - Generazione delle sequenze	6
2.1 - Import delle triple	7
2.2 - Random Walk	8
3 - Propositionalization	9
3.1 - Word2vec	10
3.1.1 - CBOW	10
3.1.2 - Skip-Gram	11
3.2 - Configuration	12
4 - Evaluation	12
4.1 - Modelli	12
4.1.1 - Classification	12
4.1.2 - Clustering	14
4.1.3 - Multiclass Classification	15
4.2 - Metriche	16
4.2.1 - Classification	16
4.2.2 - Clustering	17
4.3 - Results	18
4.3.1 - Freebase	18
4.3.2 - DBpedia	28
4.4 - Conclusioni	32

1 - Il Web Semantico

Il Web Semantico nasce dal bisogno di fornire una struttura al Web tradizionale, sia al livello più basso, ovvero quello dei dati, ma anche fornendo uno schema di livello superiore, un livello ontologico, estendendo l'idea di utilizzare schemi per descrivere domini di informazione. Dei metadati devono mappare i dati rispetto a classi, o concetti, di questo schema di dominio. In questo modo è possibile disporre di strutture in grado di descrivere e automatizzare i collegamenti esistenti fra i dati. Il web semantico deve comporsi di tre livelli fondamentali. Al livello più basso abbiamo i dati, i metadati riportano questi dati ai concetti di uno schema, nello schema (chiamato ontologia) si esprimono le relazioni fra concetti, che diventano classi di dati.

Il semantic web ha dunque l'obiettivo di dare alle informazioni un significato, correlarle tra loro e rendere le informazioni *machine-readable*. Per mettere in pratica questo concetto è necessario un modello, una tecnologia che permetta di identificare le risorse, definirne il significato e le relazioni.

Per rendere gli statement *machine-readable* sono necessari due elementi:

- Un sistema di *identificatori* per rendere soggetto, predicato e oggetto identificabili univocamente
- Un linguaggio *machine-readable* per rappresentare la frase e scambiarla tra le applicazioni.

Per identificare entità non presenti sul web viene utilizzato un identificatore più generico fornito dal web: l'**URI** (di cui gli URL sono un sottoinsieme), che ha la proprietà di poter essere creato da utenti per identificare le risorse, sia accessibili dalla rete che non.

Lo standard W3C per la rappresentazione delle informazioni delle risorse sul web è il **Resource Description Framework (RDF)**, il quale fornisce quindi un framework comune per esprimere i significati delle informazioni, ma anche per poterle renderle

condivisibili tra le applicazioni, facendo sì che i dati possano essere disponibili anche per applicazioni diverse da quelle per cui erano stati originariamente creati.

Questo linguaggio, inteso per rappresentare metadati delle risorse sul web (ad esempio autore, titolo o data di creazione di una pagina web), può anche essere generalizzato per rappresentare informazioni riguardo entità che non sono necessariamente recuperabili o scaricabili dal web (pagine web, file, video, ecc...), ma anche entità che possono essere semplicemente identificate sul web (come ad esempio persone, eventi o informazioni riguardanti prezzi di prodotti su un negozio on-line), rendendo di fatto rappresentabile qualsiasi elemento.

L'RDF sfrutta gli identificatori Web **URIs (Uniform Resource Identifiers)** per identificare gli elementi e per descriverli fa uso di proprietà e relativi valori associati. Con questi principi di base, l'RDF può rappresentare definizioni delle risorse in forma di **grafi**, formati da nodi e archi che rappresentano le risorse, le loro proprietà e i rispettivi valori. Questo modello a grafi può essere codificato in formati differenti, tramite i quali diventa possibile per le macchine elaborare il modello e comprendere il significato delle descrizioni delle risorse.

Per descrivere le risorse in RDF, vengono costruite degli *statement* composti da:

- **soggetto**, la parte della frase che identifica l'entità descritta
- **predicato**, la parte che identifica la proprietà (della entità) che viene specificata dalla frase
- **oggetto**, la parte che identifica il valore della proprietà

Gli statement possono essere visti come archi in un grafo, che sono così rappresentati:

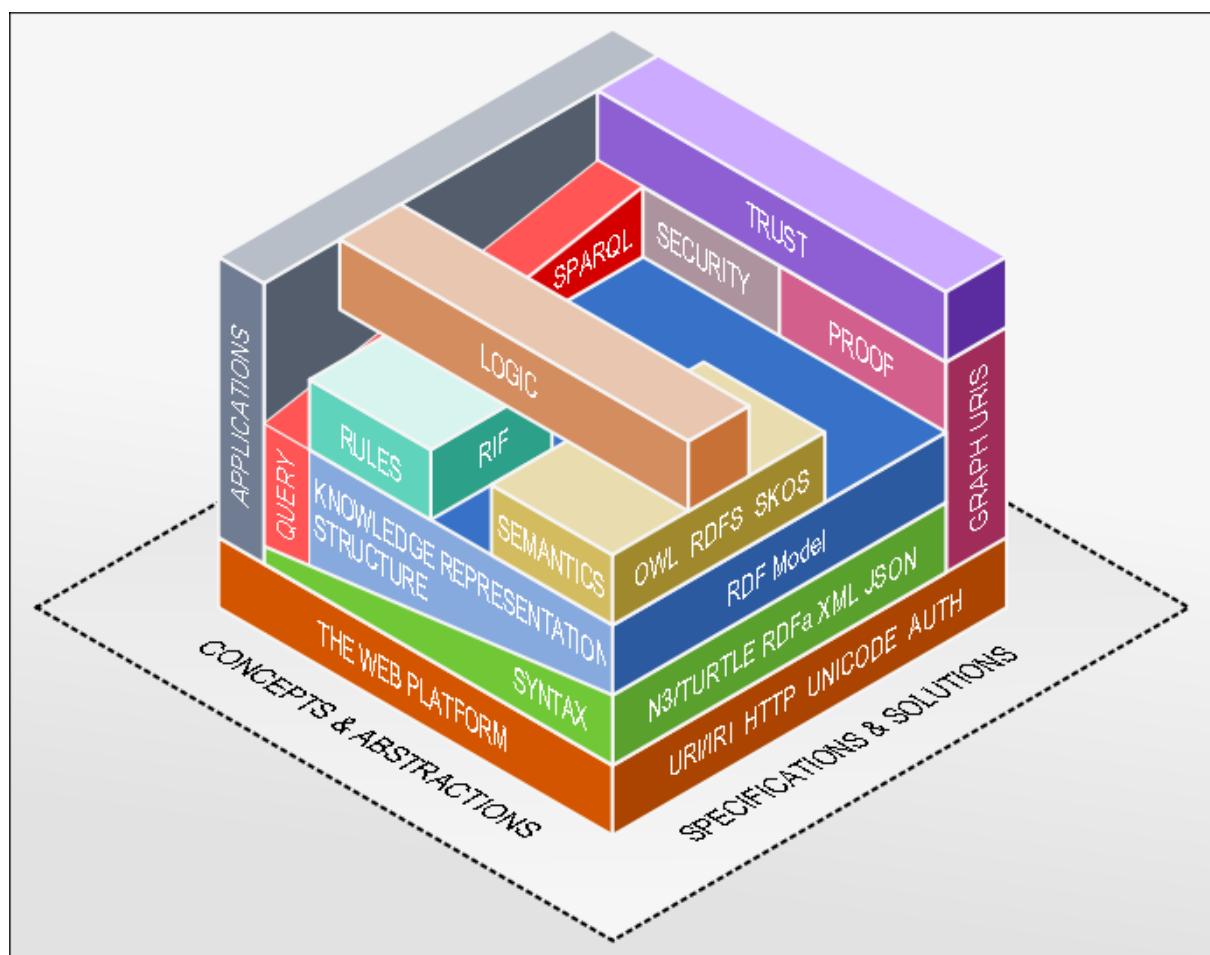
- nodo per il soggetto
- nodo per l'oggetto
- arco per il predicato, che unisce soggetto a oggetto

Gli oggetti (che rappresentano i valori associati ai prediciati) possono essere rappresentati sia tramite URI, sia come stringhe di caratteri (dette **literals**). L'utilizzo

degli URI per soggetti, predici e oggetti è importante per lo sviluppo e l'uso di **vocabolari condivisi** e ad esempio per permettere la scrittura di applicazioni che si comportano in un certo modo in base al significato che viene ricavato dall'elaborazione di un certo URI.

Il modello di rappresentazione più diffuso per le frasi RDF resta quello delle **triple**, in cui ogni frase è scritta come una tripla ordinata formata dai 3 elementi base: soggetto, predicato e oggetto, precisamente nella forma: <soggetto> <predicato> <oggetto> oppure <soggetto> <predicato> “oggetto”, se l'oggetto è literal

RDF, trattato qui sono nei suoi aspetti base, è solo uno degli elementi dell'architettura del semantic web, che possono essere rappresentati dal **layer-cake diagram** qui di seguito.



Il Semantic Web è ormai una risorsa preziosa per processi di Data Mining e più ampiamente di Knowledge Discovery. Molti tool di Data Mining però necessitano attributi in forma proposizionale, i.e. un vettore di caratteristiche numeriche o nominali (feature vector), mentre gli statement si presentano sotto forma di grafi. Di seguito verrà descritto un approccio che sfrutta tecniche di language modeling per estrarre feature in modo non supervisionato da sequenze di parole, e verrà adattato ai grafi RDF.

Il progetto si divide in tre fasi:

- Generazione delle sequenze sfruttando informazioni locali derivate da sotto-strutture del grafo RDF.
- Apprendimento delle rappresentazioni vettoriali delle entità.
- Valutazione con tecniche di Data Mining dei vettori ottenuti da Knowledge Graph generici come DBpedia e Freebase.

2 - Generazione delle sequenze

Lavori recenti hanno esteso tecniche di rappresentazione distribuita delle parole in un corpus di documenti, tradizionalmente utilizzate solamente nel Natural Language Processing, ad altri tipi di dati, incluso entità in grandi Knowledge Graph come DBpedia. Un approccio diffuso consiste nel convertire grafi in insiemi di sequenze di nodi attraverso l'esecuzione di random walk od usando graph kernel. I cammini nel grafo così creati sono utilizzati per estrarre *embedding* delle entità, come per le parole in un testo. Nel Semantic Web, il dominio è espresso espresso come fatti espressi manualmente. Sono stati omessi i letterali in quanto ritenuti non rilevanti ai fini dell'analisi.

2.1 - Import delle triple

Le relazioni binarie possono essere espresse in triple e serializzate in diversi formati. Una **tripla RDF** è un predicato binario espresso in una forma testuale, seguendo un ordine e una sintassi predefinita. Ogni tripla è caratterizzata dalla sequenza "soggetto - predicato - oggetto", tra loro separate, e termina con il simbolo del punto (.). In questo modo è possibile rappresentare testualmente multigrafi.

Le serializzazioni più diffuse sono:

- Turtle (Terse RDF Triple Language) si possono dichiarare prefissi che si ripetono negli URI, una volta dichiarati, gli URI si possono abbreviare (*CURIE*) anteponendo alla parte finale dello URI il prefisso

EXAMPLE 1

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

<#green-goblin>
    rel:enemyOf <#spiderman> ;
    a foaf:Person ;      # in the context of the Marvel universe
    foaf:name "Green Goblin" .

<#spiderman>
    rel:enemyOf <#green-goblin> ;
    a foaf:Person ;
    foaf:name "Spiderman", "Человек-паук"@ru .
```

- N-Triples ogni frase è esattamente una tripla RDF, hanno quindi più ridondanza rispetto turtle, ma sono parserizzabili linea per linea, utile quando l'intero file non entra in memoria, e sono molto comprimibili.

EXAMPLE 2

```
<http://example.org/#spiderman> <http://www.perceive.net/schemas/relationship/enemyOf> <http://example.org/#green-goblin> .
```

- RDF/XML: sintassi utilizzata per rappresentare grafi RDF come documenti XML

EXAMPLE 3

```
Complete description of all graph paths

<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
    <ex:editor>
        <rdf:Description>
            <ex:homePage>
                <rdf:Description rdf:about="http://purl.org/net/dajobe/">
                    </rdf:Description>
                </ex:homePage>
            </rdf:Description>
        </ex:editor>
    </rdf:Description>

    <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
        <ex:editor>
            <rdf:Description>
                <ex:fullName>Dave Beckett</ex:fullName>
            </rdf:Description>
        </ex:editor>
    </rdf:Description>

    <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
        <dc:title>RDF 1.1 XML Syntax</dc:title>
    </rdf:Description>
```

2.2 - Random Walk

Per poter applicare approcci di language modeling è necessario quindi costruire delle sequenze, ovvero dei cammini nel grafo, che saranno considerate come frasi. Queste saranno poi utilizzate per apprendere rappresentazioni vettoriali per ogni entità e predicato del grafo RDF.

Per catturare le informazioni locali relative alla sotto-struttura del grafo nei pressi di una specifica entità, è necessario analizzare gli archi ed i nodi entro una certa lunghezza.

Un grafo RDF è un grafo $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, dove \mathbf{V} è l'insieme dei vertici, ed \mathbf{E} è l'insieme degli archi orientati.

Per ogni vertice $v \in \mathbf{V}$ vengono generati n cammini nel grafo di lunghezza d con radice in v . Partendo dalla radice si segue casualmente un arco uscente fra quelli disponibili creando la sequenza $v \rightarrow e \rightarrow v'$, se la lunghezza del cammino è inferiore a d si continua scegliendo un arco a caso fra quelli disponibili in v' . La sequenza termina se non ci sono più archi uscenti o se si è raggiunta la lunghezza massima d .

L'insieme finale delle sequenze è l'unione di tutte le sequenze generate da tutti i vertici $v \in \mathbf{V}$ per il grafo dato \mathbf{G} .

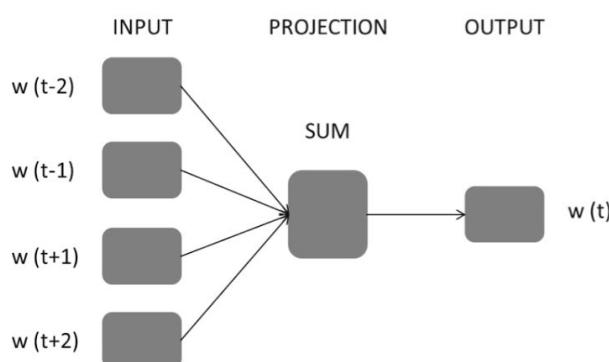
3 - Propositionalization

Negli ultimi anni sono state sviluppate tecniche di rappresentazione distribuita delle parole come alternativa alla rappresentazione come Bag of Words, dove ciascuna dimensione del vettore rappresenta una parola. Infatti, mentre questi approcci sono semplici e robusti, soffrono di alcuni svantaggi, come ad esempio l'elevata dimensionalità e sparsità, che ne limita dunque le performance. Per superare queste limitazioni, sono state proposte tecniche di apprendimento che estraggono vettori distribuiti (**embedding**) a bassa dimensionalità tramite reti neurali. L'obiettivo di tale approccio è quello di stimare la similarità di una specifica sequenza di parole in un corpus di documenti, modellando esplicitamente l'assunzione che parole vicine sono statisticamente dipendenti, e che quindi condividono parte del significato. Fra i più popolari ed utilizzati algoritmi di neural language model è Word2vec.

3.1 - Word2vec

Inizialmente proposto nel 2013 nell'articolo *"Efficient estimation of word representations in vector space"* di T. Mikolov et al., è una rete neurale a due strati molto efficiente per l'apprendimento di word embedding da testo non strutturato. Offre due algoritmi differenti , il modello Continuous Bag-of-Words (CBOW) ed il modello Skip-Gram.

3.1.1 - CBOW



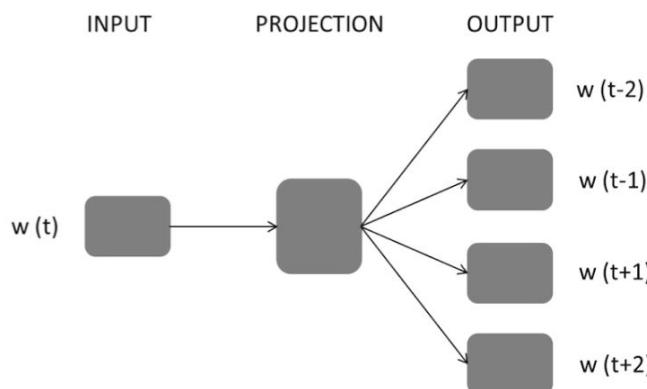
Il modello CBOW predice la parola obiettivo utilizzando le parole vicine entro una certa finestra (contesto). L'input layer è composto da tutte le parole nel contesto della parola da predire, la rete neurale quindi produce uno score per ogni parola nel vocabolario, indicando la probabilità che quella parola sia quella obiettivo. Formalmente, data una sequenza di parole $w_1, w_2, w_3, \dots, w_T$ ed un contesto c l'obiettivo del modello è massimizzare la probabilità (average log probability):

$$\frac{1}{T} \sum_{t=1}^T \log p(w_t | w_{t-c} \dots w_{t+c})$$

dove la probabilità $p(w_t | w_{t-c} \dots w_{t+c})$ è calcolata usando la funzione softmax.

$$p(w_t | w_{t-c} \dots w_{t+c}) = \frac{\exp(\bar{v}^T v'_{w_t})}{\sum_{w=1}^V \exp(\bar{v}^T v'_{w_t})}$$

3.1.2 - Skip-Gram



Il modello skip-gram fa l'opposto, cerca di predire il contesto partendo da una parola. Formalmente, data una sequenza di parole di training $w_1, w_2, w_3, \dots, w_T$, ed un contesto c , l'obiettivo è massimizzare la probabilità:

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)$$

dove la probabilità $p(w_{t+1}|w_t)$ è calcolata utilizzando sempre la funzione softmax:

$$p(w_o|w_i) = \frac{\exp(v_{wo}' v_{wi})}{\sum_{w=1}^V \exp(v_w' v_{wi})}$$

In tutti i casi, calcolare la softmax è computazionalmente inefficiente, in quanto il costo è proporzionale alla dimensione del vocabolario. Sono state proposte due tecniche: hierarchical softmax e negative sampling. Studi empirici mostrano che nella maggior parte dei casi il negative sampling porta ad una migliore performance, ma ha un tempo di runtime più elevato.

Una volta finito il training, le parole (o le entità) sono proiettate in uno spazio a bassa dimensionalità, dove parole semanticamente simili sono posizionate vicino.

3.2 - Configuration

In seguito viene utilizzato il dataset delle sequenze generate per apprendere entrambi i modelli CBOW e Skip-Gram con i seguenti parametri: window size = 5; number of iterations = 5; negative sampling; negative samples = 5. I parametri rimanenti sono stati lasciati al valore di default.

4 - Evaluation

L'approccio proposto viene valutato su task di classificazione e clustering, comparando i risultati attraverso l'utilizzo di differenti algoritmi di apprendimento.

4.1 - Modelli

4.1.1 - Classification

Random Forest

Meta-estimatore che utilizza diversi alberi di decisione appresi su diversi sotto-insiemi del dataset e combina i risultati per contrastare l'over-fitting. Metodo di Ensemble Learning, utilizzato per combinare predizioni di diversi classificatori basati su un unico algoritmo di apprendimento, finalizzato a migliorare la generalizzazione e la robustezza rispetto ad un classificatore singolo.

Ogni albero di decisione viene appreso utilizzando un sotto-insieme del training set. Quindi, nei punti di decisione dell'albero (i nodi) la divisione è scelta come la migliore divisione fra le feature presenti nel sotto-insieme casuale e non nell'intero dataset di training. Questo porta ad un aumento del bias nel complesso, ma mediando la varianza diminuisce portando un miglioramento delle prestazioni rispetto al classificatore singolo.

RBF SVM

Le **Support vector machines (SVMs)** sono un insieme di metodi di apprendimento supervisionato utilizzati per la classificazione, regressione ed outlier detection. Sono molto efficaci con vettori a dimensionalità elevata e quando il numero di dimensioni è superiore del numero degli esempi. possono essere utilizzati per dividere linearmente lo spazio vettoriale o utilizzare metodi kernel.

Il kernel **Radial Basis Function** (RBF) necessita di due parametri *gamma* e *c*. Intuitivamente *gamma* definisce quanto arriva lontano l'influenza di un singolo esempio di training, con valori bassi intesi come "lontano" e alti come "vicino". Quando *gamma* è troppo piccolo, il modello tende a non catturare la forma dei dati.

Il parametro *c* cerca di ridurre la misclassificazione degli esempi di training scegliendo la superficie di decisione. A bassi valori corrispondono superfici più lisce e regolari, mentre ad alti valori di *c* si tende a classificare correttamente tutti gli esempi di training.

AdaBoost

Meta-estimatore che inizia apprendendo un classificatore sull'intero dataset, e continua apprendendo nuove copie dello stesso classificatore (decision tree) sullo stesso dataset ma con maggior enfasi sulle istanze classificate incorrettamente, specializzandosi sui casi difficili.

Spesso definito il miglior classificatore out-of-the-box.

4.1.2 - Clustering

DBSCAN

Algoritmo basato sulla densità, connette regioni di punti con densità sufficientemente alta. DBSCAN usa una definizione di cluster basata sulla nozione di *density-reachability*. Un punto *q* è direttamente raggiungibile da un punto *p* se la loro distanza è minore di un assegnato *eps* (cioè, è parte del suo *eps*-vicinato) e se *p* è circondato da un sufficiente numero di punti, allora *p* e *q* possono essere considerati parti di un cluster. Il punto *q* è *density-reachable* da *p* se c'è una sequenza *p₁, ..., p_n* di punti con *p₁ = p* e *p_n = q* dove ogni *p_{i+1}* è density-reachable direttamente da *p_i*. Si osservi che la relazione density-reachable non è simmetrica dato che *q* potrebbe situarsi su una periferia del cluster, avendo un numero insufficiente di vicini per considerarlo un elemento genuino del cluster. Di conseguenza la nozione

density-connected diventa: due punti p e q sono density-connected se c'è un punto o tale che sia o e p sia o e q sono density-reachable.

Un cluster, che è un sotto-insieme dei punti del database, soddisfa due proprietà:

- Tutti i punti all'interno del cluster sono mutualmente density-connected.
- Se un punto è density-connected a un altro punto del cluster, anch'esso è parte del cluster.

KMeans

L'algoritmo **K-means** è un algoritmo di clustering partizionale che permette di suddividere un insieme di oggetti in K gruppi sulla base dei loro attributi.

L'obiettivo che l'algoritmo si prefissa è di minimizzare la varianza totale intra-cluster. Ogni cluster viene identificato mediante un centroide o punto medio. L'algoritmo segue una procedura iterativa. Inizialmente crea K partizioni e assegna ad ogni partizione i punti d'ingresso o casualmente o usando alcune informazioni euristiche. Quindi calcola il centroide di ogni gruppo. Costruisce quindi una nuova partizione associando ogni punto d'ingresso al cluster il cui centroide è più vicino ad esso. Quindi vengono nuovamente calcolati i centroidi per i nuovi cluster e così via, finché l'algoritmo non converge.

Spectral clustering

Molto utile quando la struttura dei cluster individuali è altamente non-convessa o più in generale quando il centro e la misura della diffusione del cluster non bastano a descrivere completamente il cluster (quando i cluster sono innestati nel piano bi-dimensionale).

4.1.3 - Multiclass Classification

One vs One

Questa strategia consiste nell'apprendere un classificatore per ogni coppia di classi. La predizione consisterà nella classe che ha ricevuto più voti. Dato che ha bisogno di apprendere $n * (n - 1) / 2$ classificatori dove n è il numero di classi, questo metodo è solitamente più lento rispetto all'One-vs-Rest. Comunque, può essere vantaggioso utilizzarlo con algoritmi che non scalano bene con il numero degli esempi, come ad esempio algoritmi kernel. Questo perché ogni classificatore coinvolge solo un sotto-insieme dei dati mentre nell'One-vs-Rest il dataset completo viene utilizzato n volte.

One vs Rest

Conosciuta anche come One-vs-All, consiste nell'apprendere un classificatore per ogni classe. Per ogni classificatore l'apprendimento avviene utilizzando la classe scelta come esempi positive e tutte le altre come negativi. Oltre alla efficienza computazionale, un altro vantaggio è l'interpretabilità, dato che ogni classe è rappresentata da un classificatore soltanto. Questa è la strategia più utilizzata la classificazione multiclassa.

4.2 - Metriche

4.2.1 - Classification

Accuracy

Numero di predizioni corrette diviso il numero totale di predizioni, trasformato in percentuale.

Log-loss

Si tratta della funzione perdita (loss function) usata nella regressione logistica e nelle sue estensioni, come le reti neurali, definita come meno logaritmo della verosimiglianza delle label positive, date delle predizioni probabilistiche. È definita soltanto per due o più label.

Precision

È il numero di True Positive diviso per il numero di True Positive e False Positive. Può essere vista come la misura della esattezza del classificatore. Bassa precision indica alto numero di False Positive.

Recall

È il numero di True Positive diviso per il numero di True Positive e il numero di False Negative. Ovvero il numero delle predizioni positive rapportato a tutte le istanze positive nel test set. Viene chiamata anche Sensitivity. Può essere visto come la completezza del classificatore. Bassa recall indica molti False Negative.

F1-score

Media armonica tra la Precision e la Recall.

Matthews

Viene utilizzato come misura della qualità di classificatori binari. Considera True/False Positives/Negatives ed è generalmente considerata una misura bilanciata che può essere utilizzata persino se le classi sono sbilanciate. È un coefficiente di correlazione compreso fra -1 e +1, dove -1 rappresenta predizione inversa, 0 predizione media e +1 perfetta predizione.

4.2.2 - Clustering

Homogeneity

Metrica di una clusterizzazione data una ground truth.

Un clustering soddisfa l'Homogeneity se tutti i suoi cluster contengono solo istanze della stessa classe nella ground truth.

Completeness

Metrica di una clusterizzazione data una ground truth.

Un clustering soddisfa la Completeness se tutte le istanze di una classe nella ground truth sono contenute nello stesso cluster.

V-measure

Media armonica tra la Completeness e la Homogeneity.

Rand index

Calcola la similarità fra due clustering diversi, considerando tutte le coppie di esempi e contando le coppie che sono assegnate nello stesso o in un cluster differente.

Mutual information

È una misura di similarità fra due clustering diversi.

$$MI(U, V) = \sum_{i=1}^R \sum_{j=1}^C P(i, j) \log \frac{P(i, j)}{P(i)P'(j)}$$

Silhouette

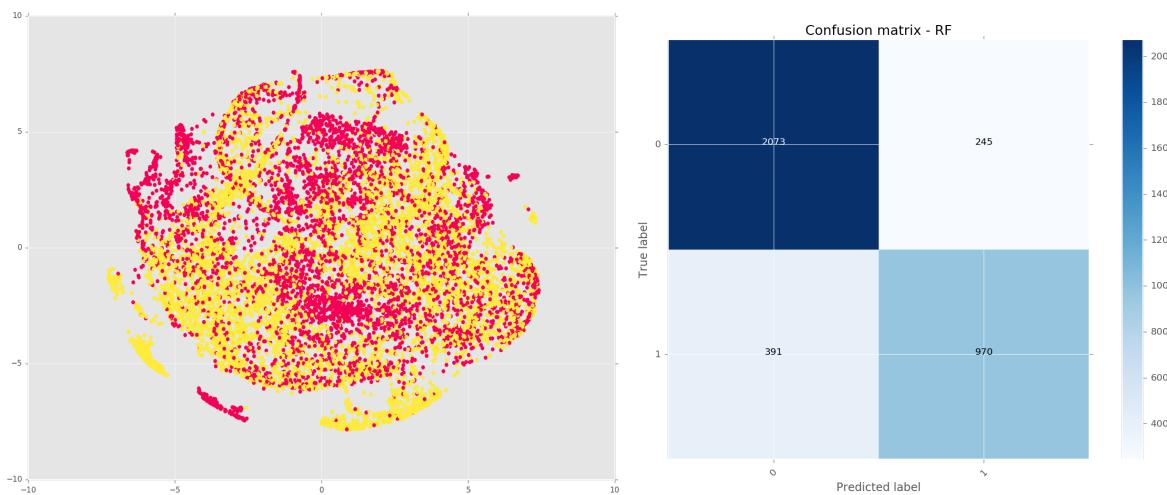
Coefficiente calcolato utilizzando la distanza media intra-cluster e la distanza media dal cluster più vicino.

È l'unico che non necessita una ground truth. Rappresenta quanto bene ogni istanza è contenuta all'interno del cluster di appartenenza.

4.3 - Results

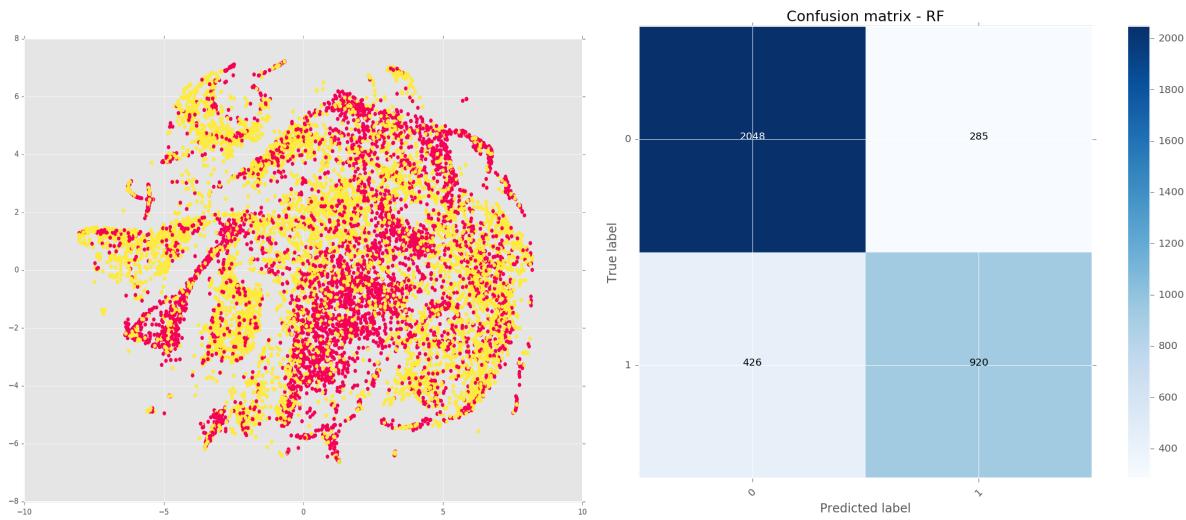
4.3.1 - Freebase

Classification **CBOW depth 5 random walks 7**



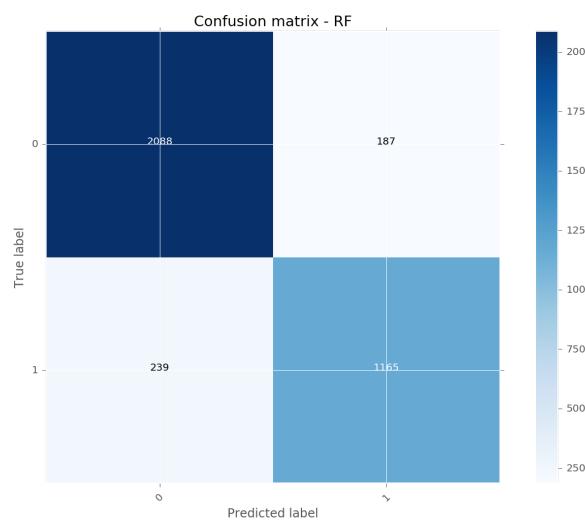
	Accuracy	Log-loss	Precision	Recall	F1-score	Matthews
Random Forest	0.827127	5.970877	0.798354	0.712711	0.753106	0.623129
Naive Bayes	0.543354	15.772337	0.444752	0.943424	0.604520	0.295864
RBF SVM	0.816798	6.327613	0.820130	0.646583	0.723090	0.598350
AdaBoost	0.819244	6.243138	0.779294	0.713446	0.744918	0.606783

Classification CBOW depth 9 random walks 20



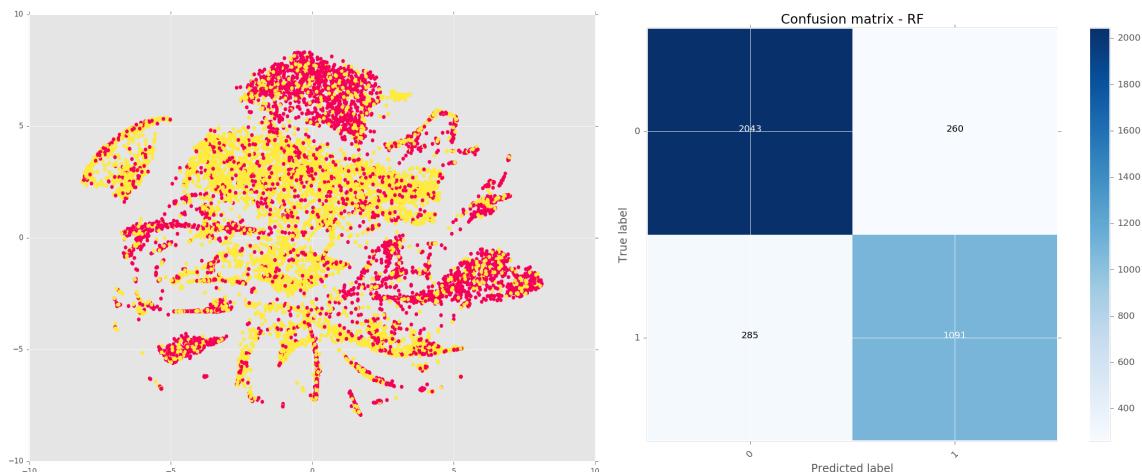
	Accuracy	Log-loss	Precision	Recall	F1-score	Matthews
Random Forest	0.806741	6.674993	0.763485	0.683507	0.721286	0.576126
Naive Bayes	0.502854	17.171181	0.416753	0.898217	0.569343	0.204027
RBF SVM	0.830932	5.839447	0.792407	0.728826	0.759288	0.630657
AdaBoost	0.794238	7.106858	0.729899	0.694651	0.711839	0.552391

Classification Skip-Gram depth 5 random walks 7



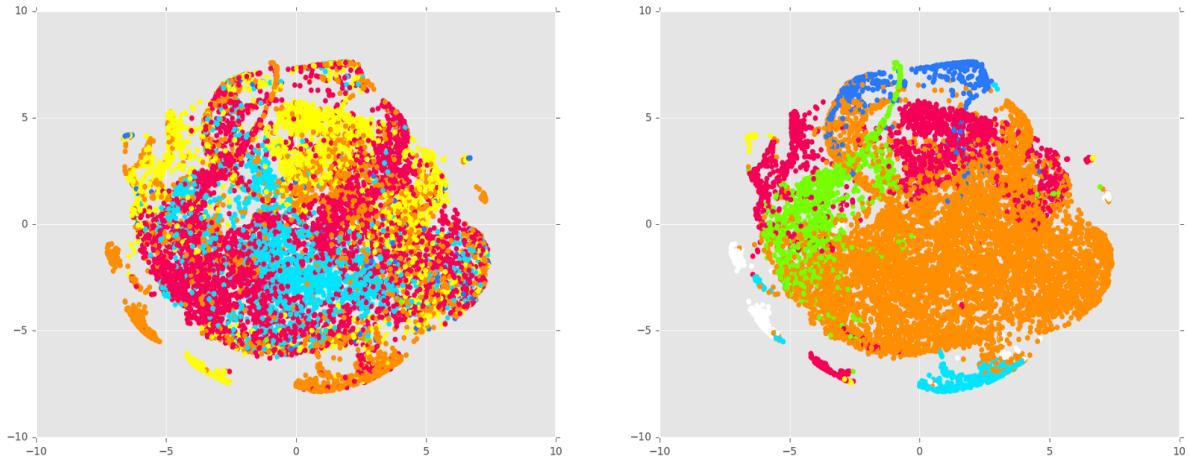
	Accuracy	Log-loss	Precision	Recall	F1-score	Matthews
Random Forest	0.884208	3.999366	0.861686	0.829772	0.845428	0.753255
Naive Bayes	0.729546	9.341340	0.593937	0.920940	0.722145	0.526166
RBF SVM	0.884479	3.989976	0.867217	0.823362	0.844720	0.753490
AdaBoost	0.868986	4.525110	0.831178	0.824074	0.827611	0.721976

Classification Skip-Gram depth 9 random walks 20



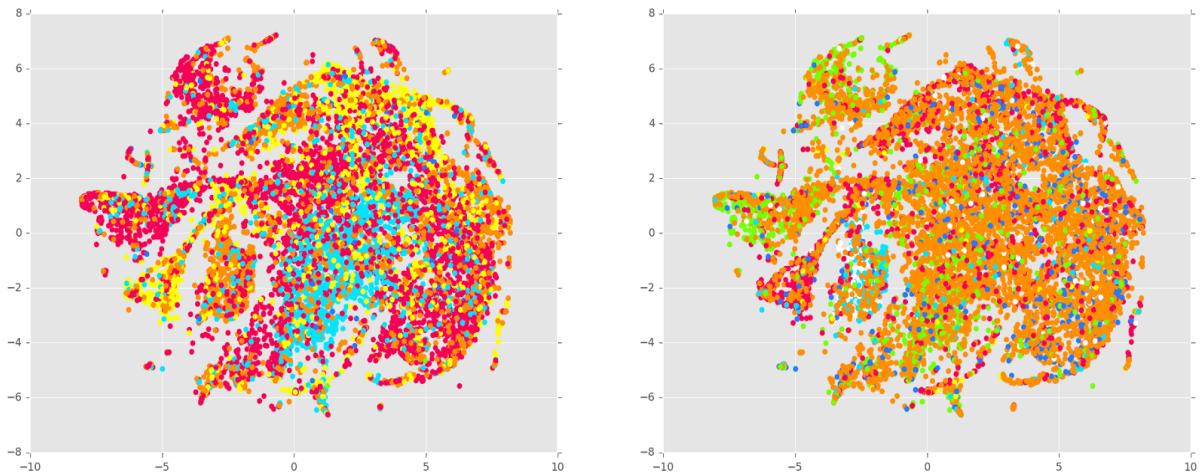
	Accuracy	Log-loss	Precision	Recall	F1-score	Matthews
Random Forest	0.851862	5.116565	0.807550	0.792878	0.800147	0.682550
Naive Bayes	0.761620	8.233497	0.637013	0.843023	0.725680	0.538094
RBF SVM	0.863822	4.703480	0.831691	0.797238	0.814100	0.707128
AdaBoost	0.834466	5.717401	0.800313	0.742733	0.770449	0.642387

Clustering CBOW depth 5 random walks 7



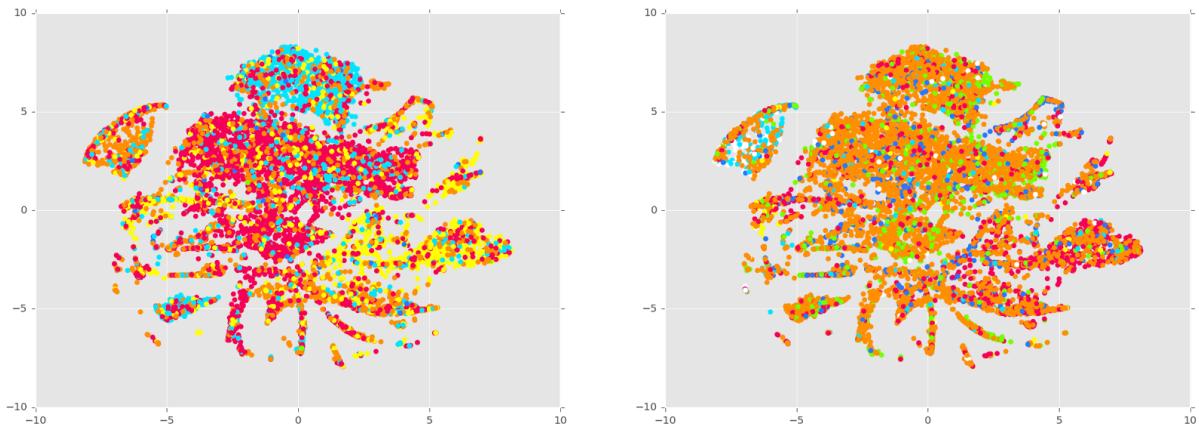
	Homogeneit y	Completeness s	V-Measure score	Adjusted Rand index	Mutual Information	Silhouette
HDBSCAN	0.005875	0.015756	0.008558	0.000728	0.005597	-0.152478
DBSCAN	0.002416	0.113639	0.004731	0.001536	0.002284	0.914751
KMeans	0.191475	0.208746	0.199738	0.102289	0.190524	0.292381
Spectral	0.195493	0.218288	0.206263	0.121244	0.194775	0.049949
Agglo	0.000601	0.143566	0.001197	0.000114	0.000091	0.945183

Clustering CBOW depth 9 random walks 20



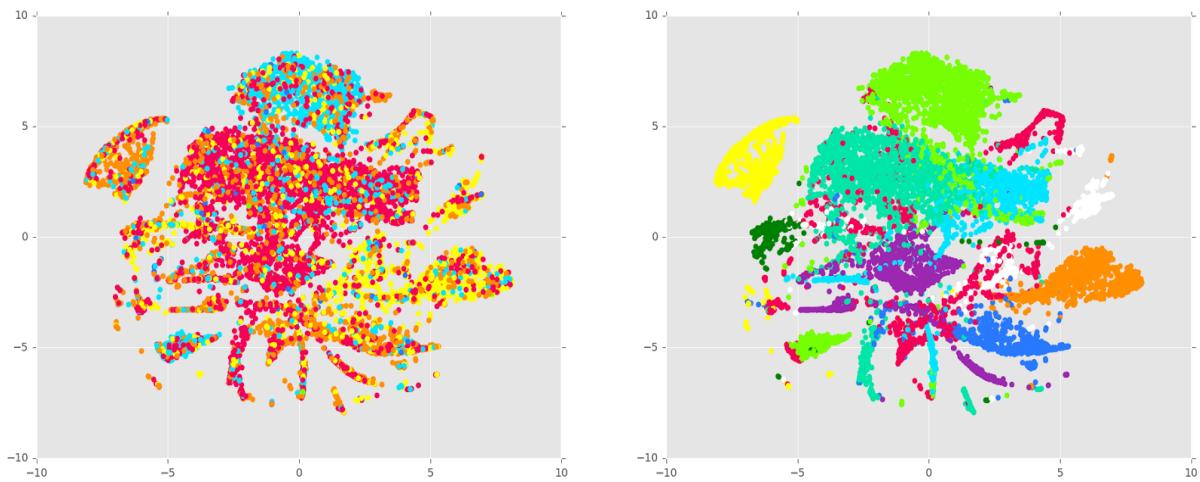
	Homogeneity	Completeness	V-Measure score	Adjusted Rand index	Mutual Information	Silhouette
DBSCAN	0.011909	0.027964	0.016704	0.020919	0.011297	0.016050
KMeans	0.161125	0.153432	0.157185	0.083499	0.152401	0.196027
Spectral	0.195491	0.218281	0.206259	0.121214	0.194774	-0.125369
Agglom	0.000601	0.143559	0.001197	0.000114	0.000091	-0.315953

Clustering Skip-Gram depth 5 random walks 7



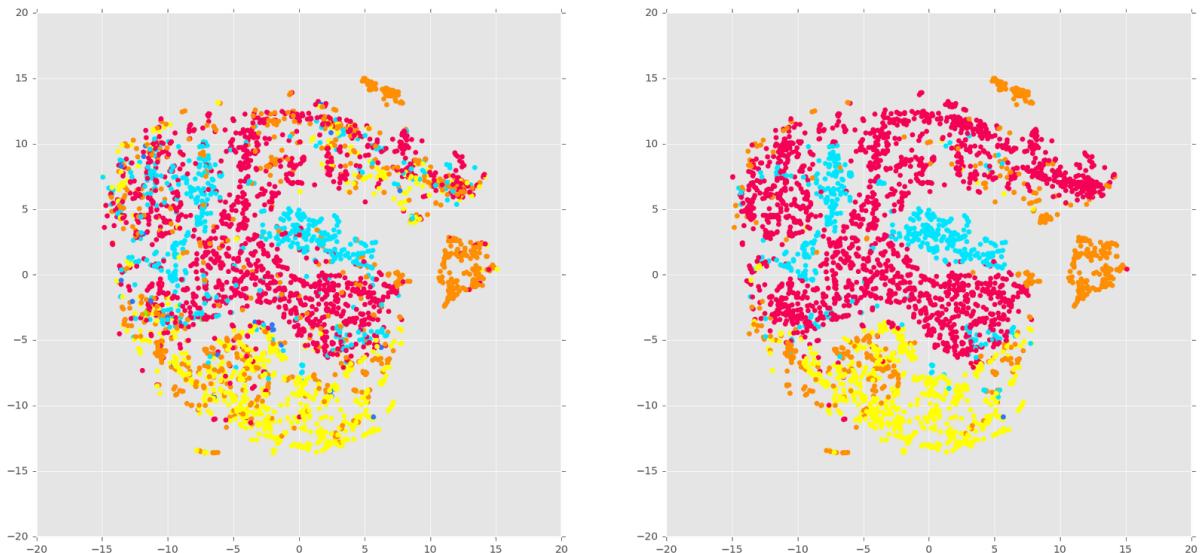
	Homogeneity	Completeness	V-Measure score	Adjusted Rand index	Mutual Information	Silhouette
DBSCAN	0.012680	0.185138	0.023734	0.007639	0.012296	0.643485
KMeans	0.354985	0.261584	0.301210	0.195248	0.260861	0.229451
Spectral	0.195493	0.218288	0.206263	0.121244	0.194775	0.090845
Agglomerative	0.000601	0.143566	0.001197	0.000114	0.000091	0.790993

Clustering Skip-Gram depth 9 random walks 20

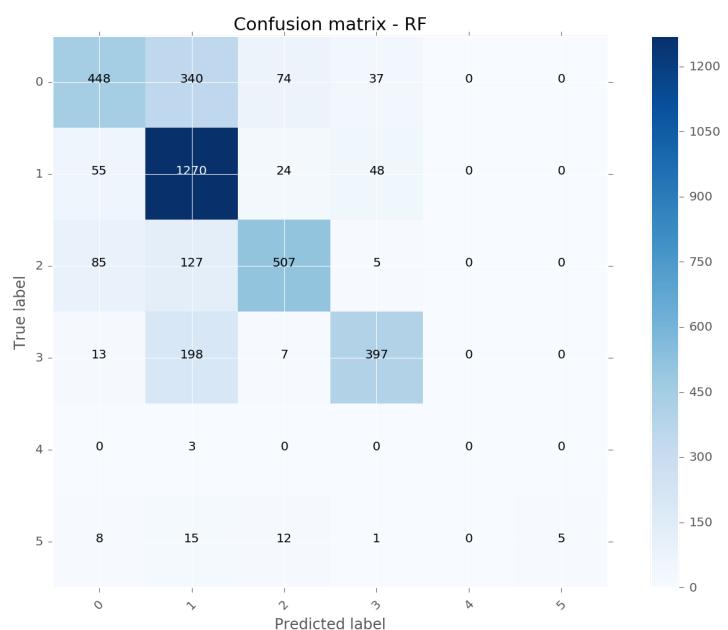


	Homogeneity	Completeness	V-Measure score	Adjusted Rand index	Mutual Information	Silhouette
DBSCAN	0.055040	0.106756	0.072633	0.031644	0.052613	-0.349706
KMeans	0.299188	0.190991	0.233148	0.174406	0.190289	0.199499
Spectral	0.195491	0.218281	0.206259	0.121214	0.194774	-0.087163
Agglomerative	0.000601	0.143559	0.001197	0.000114	0.000091	-0.185102

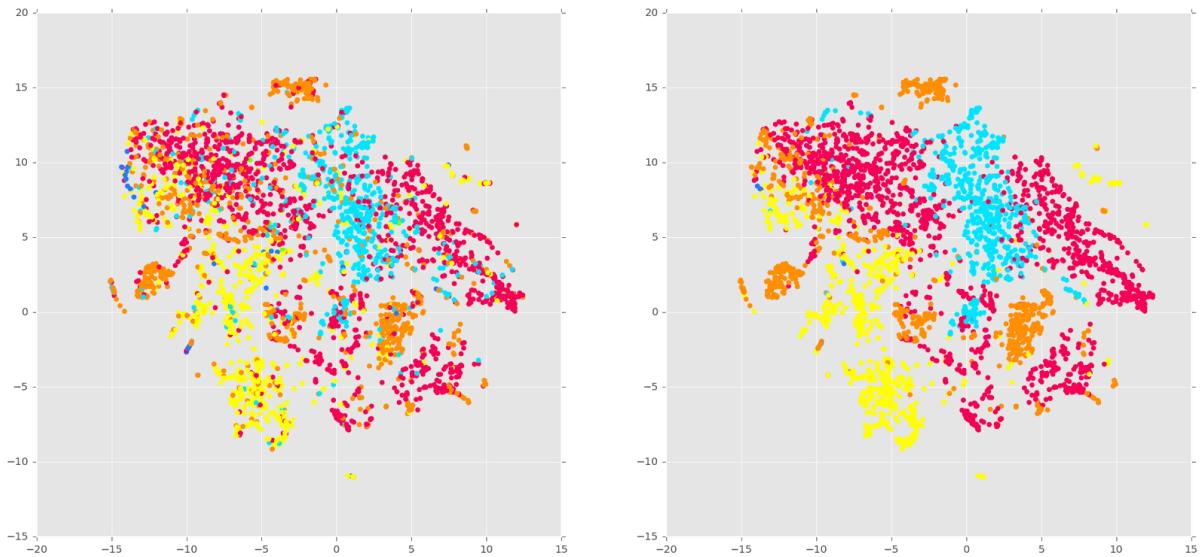
Multiclass Classification CBOW depth 5 random walks 7



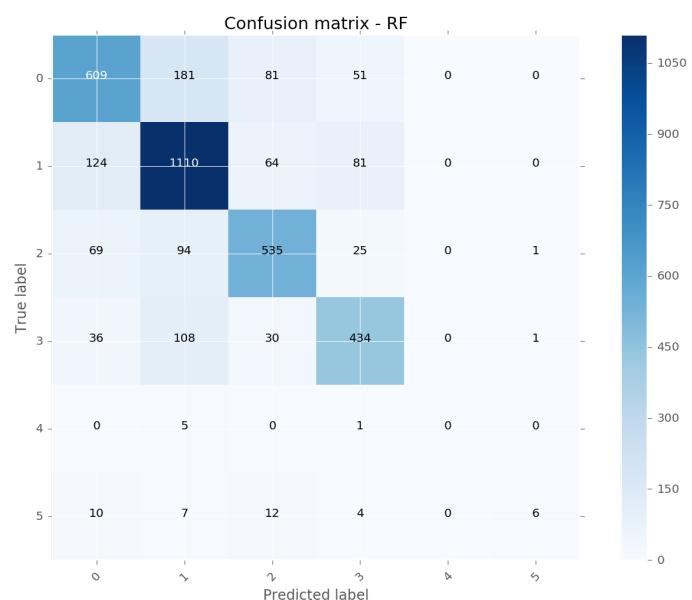
	Accuracy	Kappa	Precision	Recall	F1-score
OnevOne RBF SVM	0.714053	0.592904	0.733717	0.714053	0.703889
OnevOne Forest	0.668660	0.525398	0.688734	0.668660	0.653022
OnevRest RBF SVM	0.736613	0.631140	0.747597	0.736613	0.724743
OnevRest Forest	0.653710	0.500355	0.684843	0.653710	0.632705



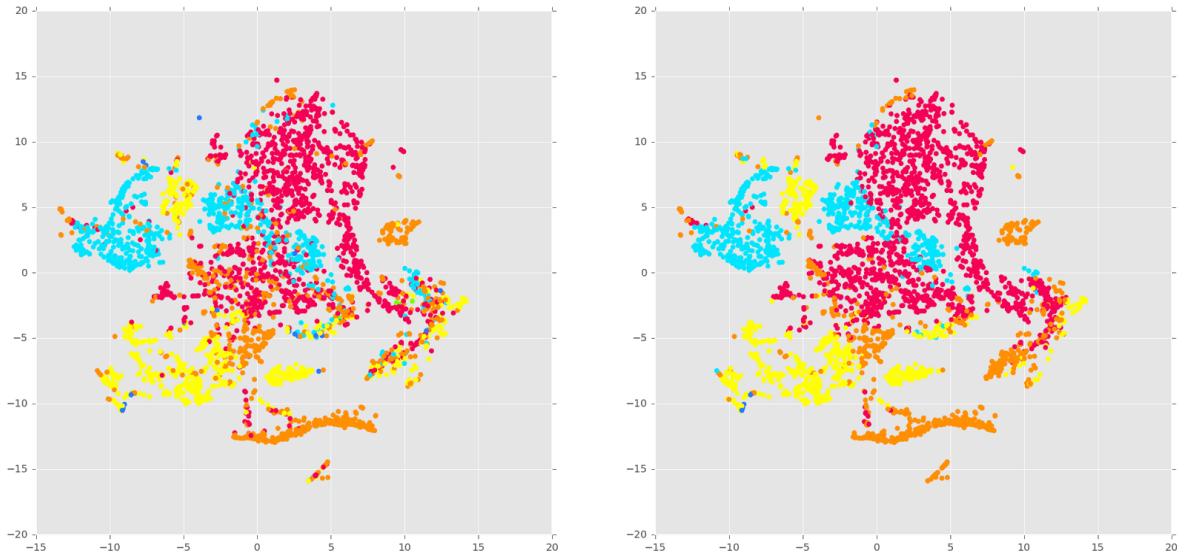
Multiclass Classification CBOW depth 9 random walks 20



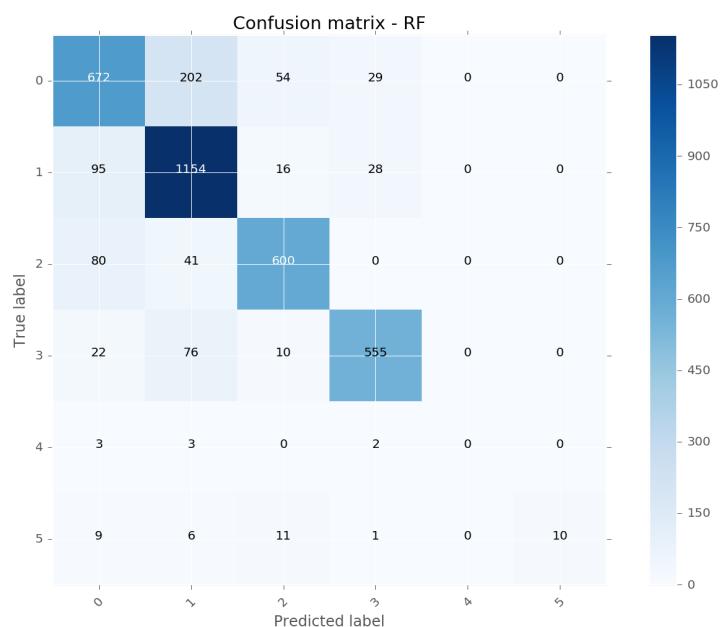
	Accuracy	Log-loss	Precision	Recall	F1-score	Matthews
Random Forest	0.827127	5.970877	0.798354	0.712711	0.753106	0.623129
Naive Bayes	0.543354	15.772337	0.444752	0.943424	0.604520	0.295864
RBF SVM	0.816798	6.327613	0.820130	0.646583	0.723090	0.598350
AdaBoost	0.819244	6.243138	0.779294	0.713446	0.744918	0.606783



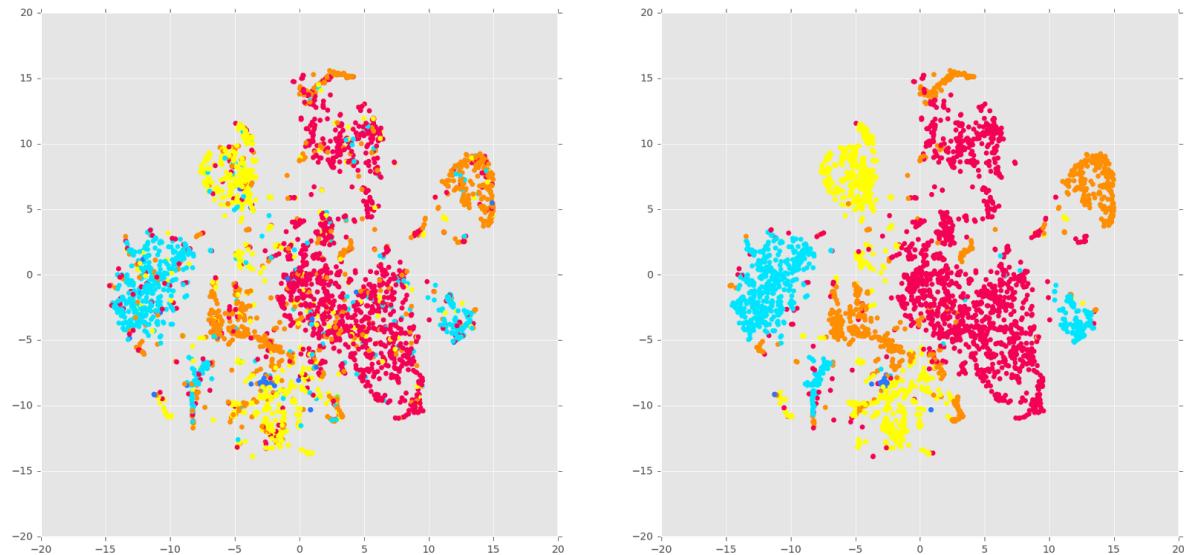
Multiclass Classification Skip-Gram depth 5 random walks 7



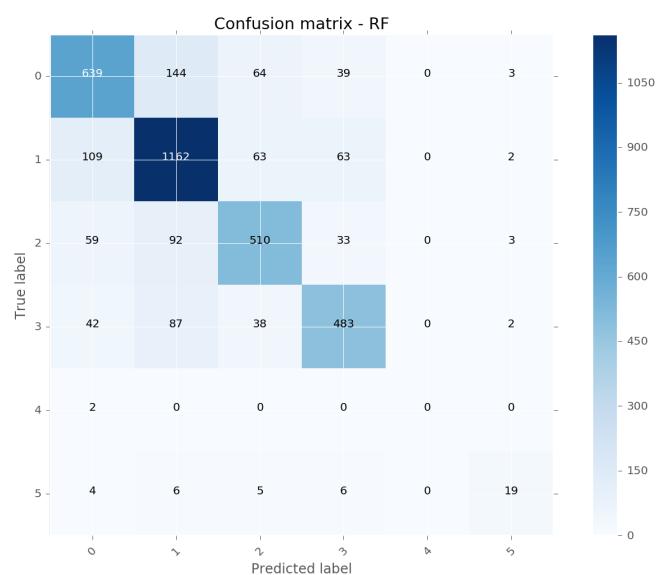
	Accuracy	Kappa	Precision	Recall	F1-score
OnevOne RBF SVM	0.812993	0.743535	0.814941	0.812993	0.809874
OnevOne Forest	0.752378	0.657869	0.760655	0.752378	0.744670
OnevRest RBF SVM	0.821691	0.755987	0.823960	0.821691	0.816511
OnevRest Forest	0.745039	0.646730	0.767992	0.745039	0.732363



Multiclass Classification Skip-Gram depth 9 random walks 20



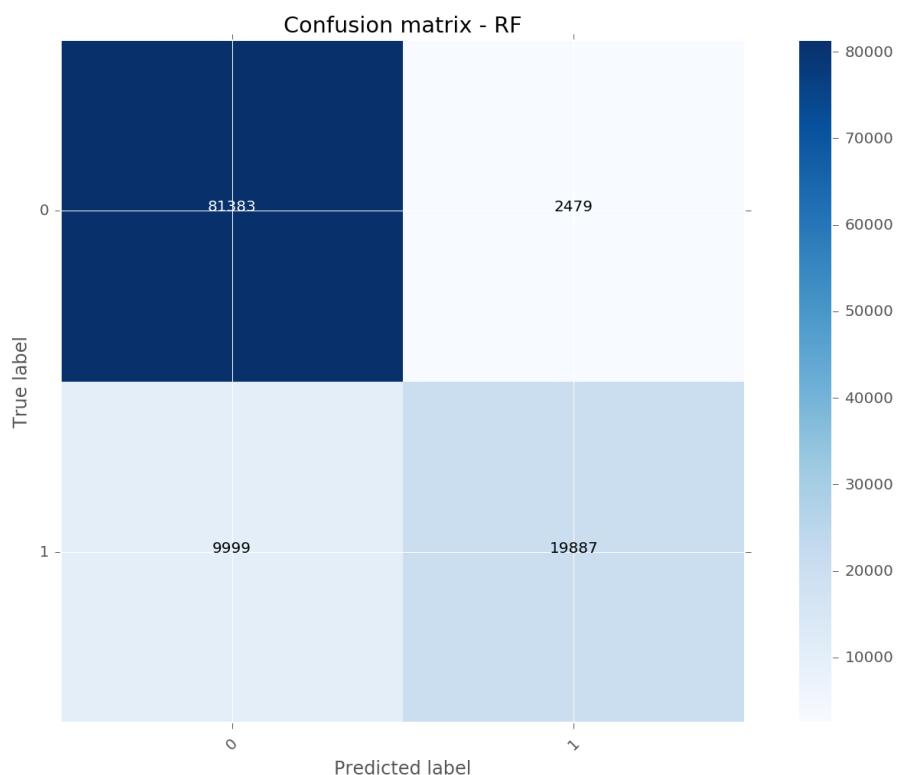
	Accuracy	Kappa	Precision	Recall	F1-score
OnevOne RBF SVM	0.764610	0.675131	0.763344	0.764610	0.763357
OnevOne Forest	0.690949	0.566118	0.689110	0.690949	0.682569
OnevRest RBF SVM	0.769231	0.682410	0.767976	0.769231	0.768194
OnevRest Forest	0.690949	0.563482	0.696911	0.690949	0.681488



4.3.2 - DBpedia

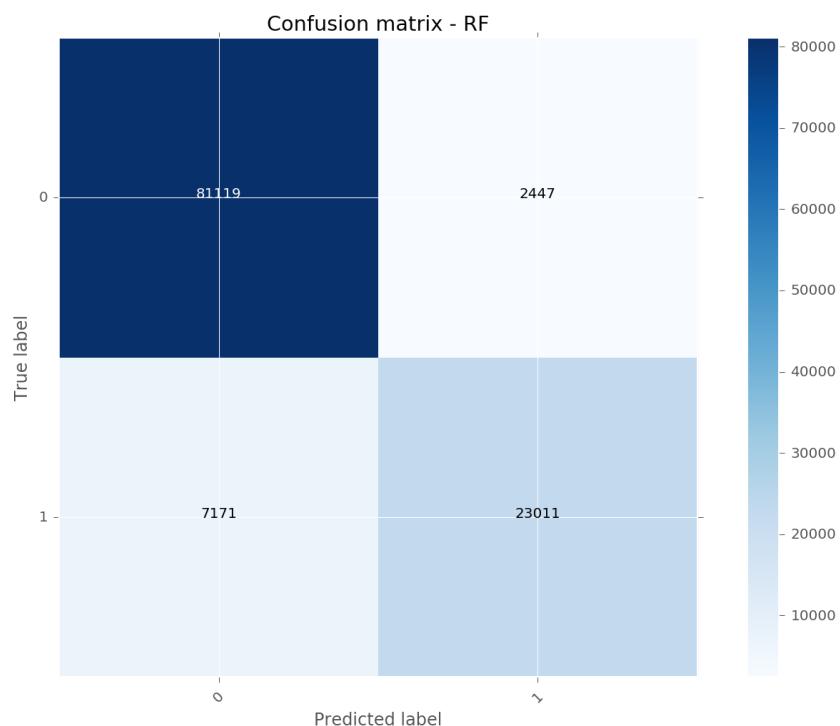
Classification **CBOW** depth **5** random walks **7**

	Accuracy	Log-loss	Precision	Recall	F1-score	Matthews
Random Forest	0.890301	3.788874	0.889162	0.665429	0.761196	0.704140
Naive Bayes	0.351655	22.393557	0.275513	0.900622	0.421947	0.071601
RBF SVM	0.879593	4.158704	0.921001	0.592552	0.721139	0.674577
AdaBoost	0.848586	5.229670	0.809110	0.554541	0.658063	0.581772



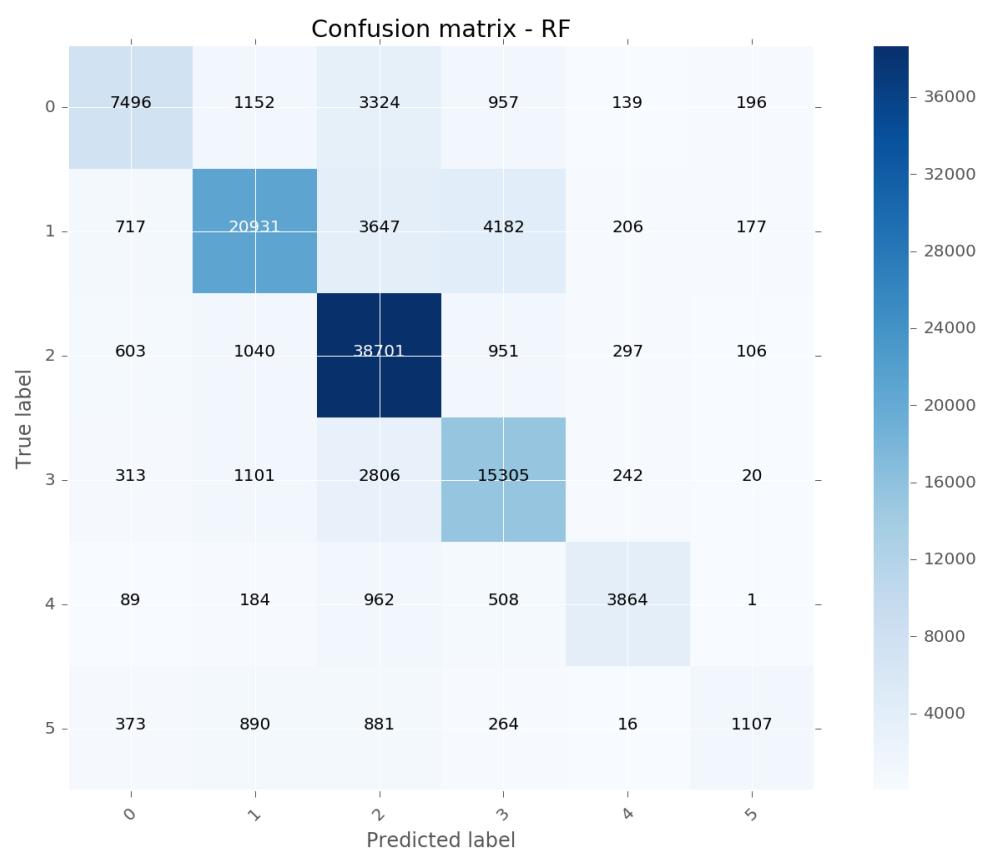
Classification Skip-Gram depth 5 random walks 7

	Accuracy	Log-loss	Precision	Recall	F1-score	Matthews
Random Forest	0.915445	2.920455	0.903881	0.762408	0.827139	0.776604
Naive Bayes	0.607905	13.542754	0.389133	0.838347	0.531542	0.323902
RBF SVM	0.918179	2.826017	0.927924	0.749884	0.829458	0.784054
AdaBoost	0.862978	4.732614	0.795537	0.650885	0.715978	0.632355



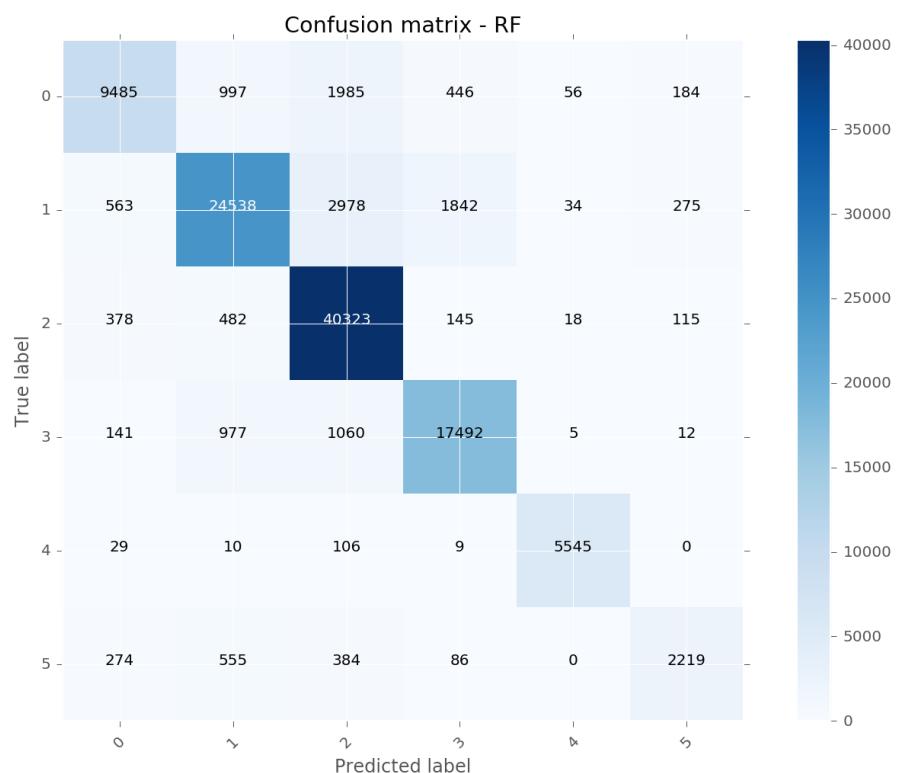
Multiclass Classification CBOW depth 5 random walks 7

	Accuracy	Kappa	Precision	Recall	F1-score
OnevRest RBF SVM	0.768400	0.684129	0.771742	0.768400	0.761103
OnevRest Forest	0.596169	0.405588	0.668737	0.596169	0.552088



Multiclass Classification Skip-Gram depth 5 random walks 7

	Accuracy	Kappa	Precision	Recall	F1-score
OnevRest RBF SVM	0.875637	0.832115	0.876130	0.875637	0.873039
OnevRest Forest	0.712391	0.593899	0.764538	0.712391	0.686503



4.4 - Conclusioni

In questo progetto è stato presentato un approccio per apprendere rappresentazioni numeriche latenti delle entità in un grafo RDF. È stato necessario convertire i grafi in insiemi di sequenze, che sono state usate per costruire neural language model. Sono stati considerati, per la valutazione dei vettori appresi, solo semplici task di machine learning, come classificazione e clustering, ma in lavori futuri le applicazioni potrebbero essere diverse. Per esempio le rappresentazioni vettoriale potrebbero essere utilizzate per costruire content-based recommender systems, utilizzati per task di link-prediction, graph completion (KGC) o anche per misurare la relazione semantica tra due entità.