



University of Padova

---

Department of Information Engineering



# WORD MOVER'S EMBEDDING:

From Word2Vec to Document Embedding

Mario Avdullaj

Ilaria Gallo

Alessio Mazzetto



## ❖ Bag of Words (BOW)

- ✓ semplice, efficiente, alta precisione
- ✗ non tiene conto della semantica
- ✗ rappresentazione sparsa con elevato numero di features

## ❖ Decomposizione ai valori singolari

- ✓ rappresentazione più significativa
- ✗ non sempre efficace

## ❖ Media pesata degli embeddings

- ✓ tiene conto del contesto
- ✗ traslascia l'ordine delle parole

## ❖ Doc2Vec

- ✓ apprendimento degli embeddings per parole e testi
- ✗ tiene conto dell'ordine delle parole solo all'interno di una piccola finestra

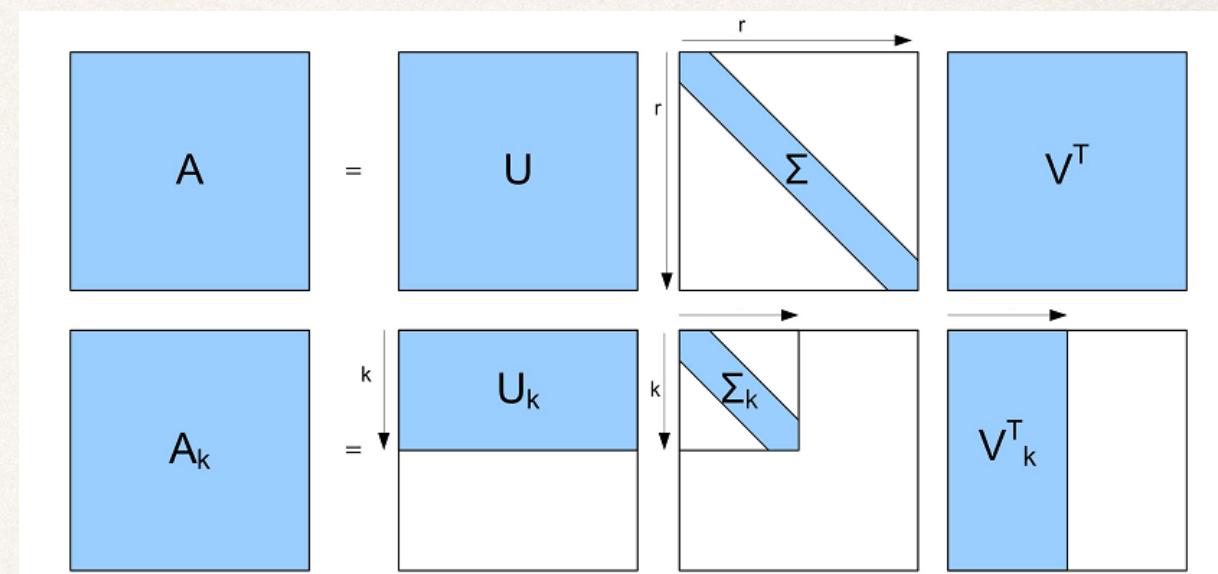


Fig.1: Decomposizione ai valori singolari

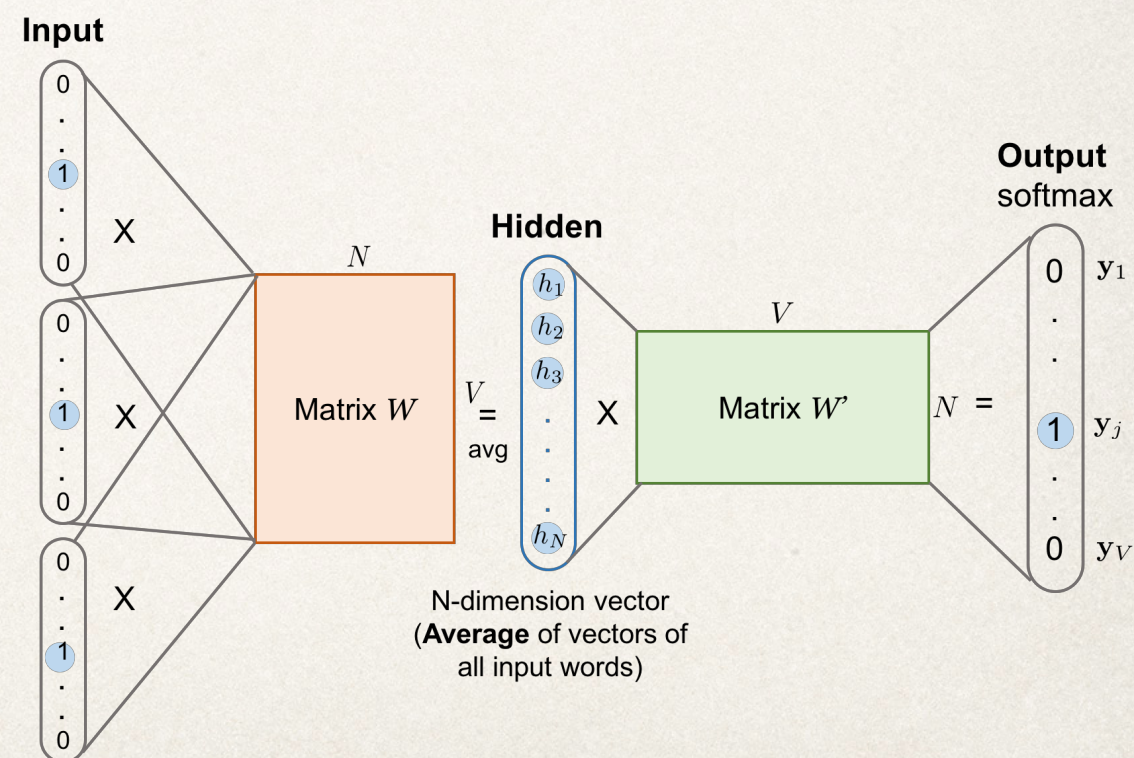
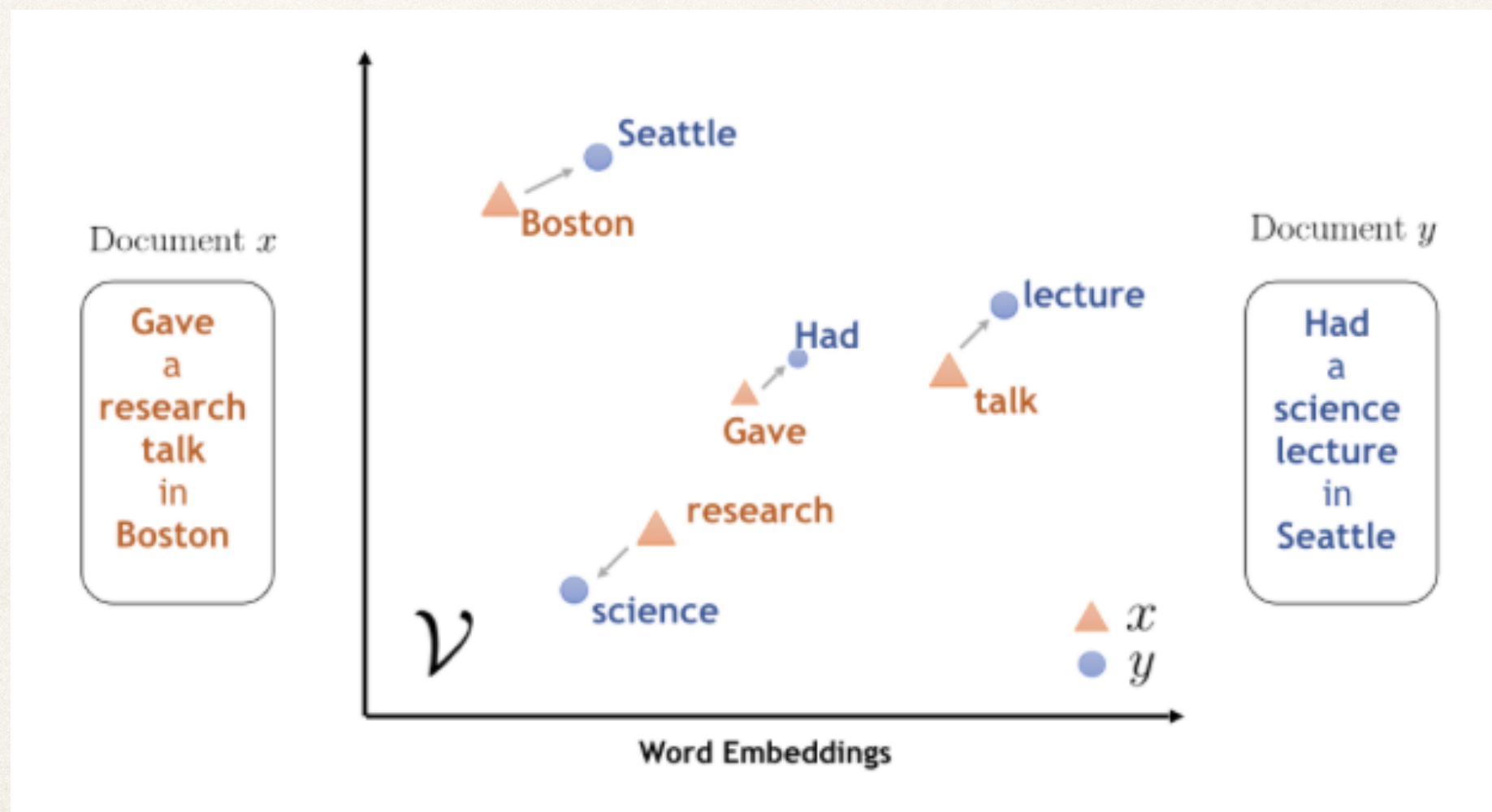


Fig.2: Word2Vec: CBOW model



**Word Mover's Distance (WMD):** misura la differenza tra due documenti testuali nello spazio degli embeddings prodotto da Word2Vec







# Word Mover's Distance

## Definizione:

$$\text{WMD}(x, y) := \min_{F \in \mathbb{R}_+^{|x| \times |y|}} \langle C, F \rangle,$$
$$\text{s.t.}, F\mathbf{1} = \mathbf{f}_x, \quad F^T\mathbf{1} = \mathbf{f}_y.$$

$x, y$ : documenti

$\mathbf{f}_x, \mathbf{f}_y$ : vettori delle  
frequenze normalizzate

$C$ : matrice dei costi

$F$ : matrice di flusso

Formulazione equivalente  
del problema di  
programmazione lineare:

$$\min_{\mathbf{T} \geq 0} \sum_{i,j=1}^n \mathbf{T}_{ij} c(i, j)$$
$$\text{subject to: } \sum_{j=1}^n \mathbf{T}_{ij} = d_i \quad \forall i \in \{1, \dots, n\}$$
$$\sum_{i=1}^n \mathbf{T}_{ij} = d'_j \quad \forall j \in \{1, \dots, n\}.$$

$d_i$ : frequenza della  
parola  $i$ -esima nel  
documento  $x$

$d_j$ : frequenza della  
parola  $j$ -esima nel  
documento  $y$





# Word Mover's Distance

- ✓ Accurato nella misura di distanza tra documenti con parole semanticamente simili ma sintatticamente differenti
- ✓ Buone performance se combinato con K-Nearest neighbors per task di classificazione
- ✗ Calcolo della distanza computazionalmente oneroso:  
 $O(L^3 \log(L))$  per una coppia di documenti  
 $O(N^2 L^3 \log(L))$  su una collezione di  $N$  documenti





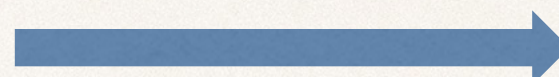
# Word Mover's Kernel



Siano  $x, y$  due documenti

$WMD(x, y)$

[1]



$k(x, y)$

Distance metric

Positive-definite kernel

[1] D2ke: From distance to kernel and embedding  
Wu et Al. (2018)

[2] Random features for large-scale kernel machine  
Ali Rahimi and Benjamin Recht. (2007)

[2]



Embedding WME





# Word Mover's Kernel

Siano  $x, y$  due documenti

$$k(x, y) := \int_{w \in \Omega} p(w) \phi_w(x) \phi_w(y) dw$$

$$\phi_w(x) = e^{-\gamma \cdot WMD(x, y)}$$

$\Omega$  = spazio dei documenti randomici

$\gamma$  = parametro di smoothing





# Word Mover's Kernel

Siano  $x, y$  due documenti

$$k(x, y) := \int_{w \in \Omega} p(w) \phi_w(x) \phi_w(y) dw$$

$$k(x, y) = e^{-\gamma \cdot \underbrace{\text{softmin}_{p(w)} [WMD(x, w) + WMD(w, y)]}_{f(w)}}$$

$$\text{softmin}_{p(w)} f(w) := -\frac{1}{\gamma} \log \int_{w \in \Omega} p(w) e^{-\gamma f(w)} dw \xrightarrow{\gamma \rightarrow +\infty} \min_{w \in \Omega} f(w)$$

$$WMD(x, y) \leq \min_{w \in \Omega} f(w)$$





# Word Mover's Kernel

## Approssimazioni:

- $\Omega$  contiene documenti con lunghezza uniformemente distribuita tra 1 e  $D_{max}$
- Ogni documento randomico è costituito da parole  $\mathbf{u} \in R^d$ , con  $u_j \sim U[v_{min}, v_{max}]$ 
  - $d$  = dimensione rappresentazione distribuita usata per i termini
  - $v_{min}, v_{max}$  = valori minimi e massimi delle coordinate dei termini nel vocabolario



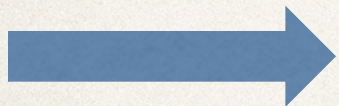


# Word Mover's Embedding

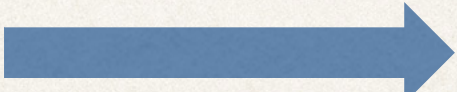
Supponendo che  $\{\omega_i\}_{i=1}^R$  siano documenti estratti casualmente

$$\begin{aligned} k(x, y) &:= \int_{w \in \Omega} p(w) \phi_w(x) \phi_w(y) dw \\ &= e^{-\gamma \cdot \text{softmin}_{p(w)} f(w)} \simeq e^{-\gamma \cdot \text{WMD}(x, y)} \end{aligned}$$

Approssimazione



$$k(x, y) \simeq \frac{1}{R} \sum_{i=1}^R \phi_{w_i}(x) \cdot \phi_{w_i}(y) = \langle Z(x), Z(y) \rangle$$

Dove  $Z(x) = \frac{1}{\sqrt{R}} \left( \phi_{w_1}(x), \dots, \phi_{w_R}(x) \right)$   EMBEDDING DI  $x$ !





# Word Mover's Embedding

Algoritmo WME:

*Per  $i = 1, \dots, R$  :*

- Estraggo  $D_i \sim U[1, D_{max}]$
- $w_i = (u_1, u_2, \dots, u_{D_i}) : u_j \sim U[v_{min}, v_{max}]^d$
- *Per ogni documento  $x$ , calcolo  $\phi_{w_i}(x)$*

L'embedding di un documento  $x$  è

$$Z(x) = \frac{1}{\sqrt{R}} \left( \phi_{w_1}(x), \dots, \phi_{w_R}(x) \right)$$





# Word Mover's Embedding



## Convergenza dell'algoritmo:

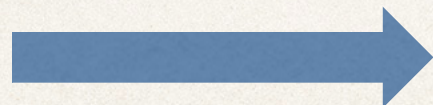
Se  $R = \Omega \left( \frac{dL}{\epsilon^2} \log \left( \frac{\gamma}{\epsilon} \right) + \frac{1}{\epsilon} \log \left( \frac{1}{\delta} \right) \right)$ , allora con probabilità almeno  $1 - \delta$ , si ha che

$$| k(x, y) - \langle Z(x), Z(y) \rangle | < \epsilon$$

## Vantaggio computazionale:

$WMD(x, y)$  richiede  $O(L^3 \log(L))$

Il calcolo di  $\phi_w(x)$  richiede  $O(D^2 L \cdot \log(L))$



Calcolo di WME super lineare in N!

$$O(N \cdot R \cdot L \cdot \log(L))$$





# Esperimenti e risultati

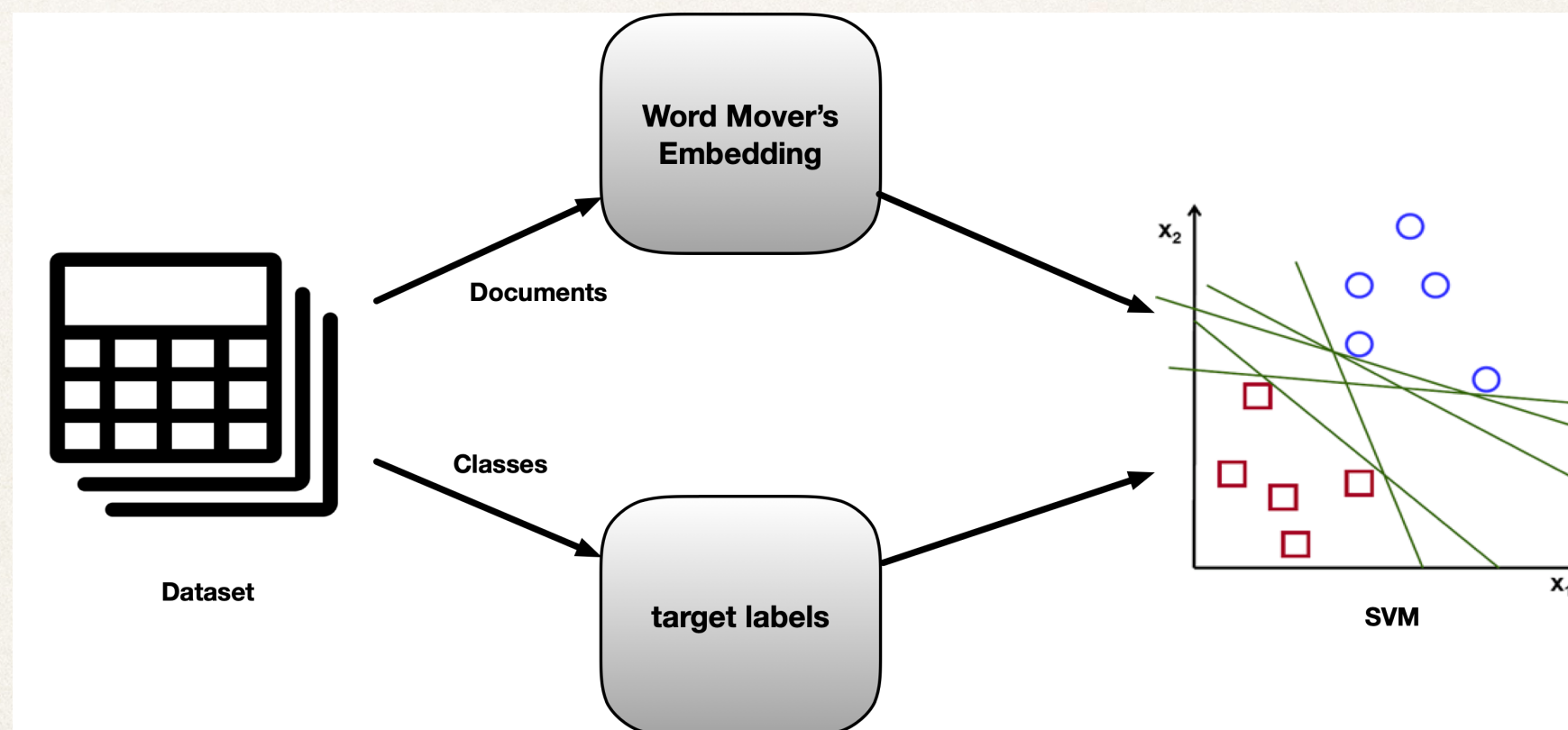


- Performance di WME variando i parametri
  - Confronto tra KNN + WMD e SVM + WME
  - Confronto tra WME e diversi metodi di Document Representation
  - Implementazione di WME in Python e risultati sperimentali
-



## Setup dell'esperimento:

- Cross-validation per la scelta di  $\gamma$  ( range di valori [0.01, 10] )
- Utilizzo di un modello di classificazione SVM lineare
- 9 Dataset utilizzati, tra cui BBCSPORT, TWITTER, 20NEWS e CLASSIC

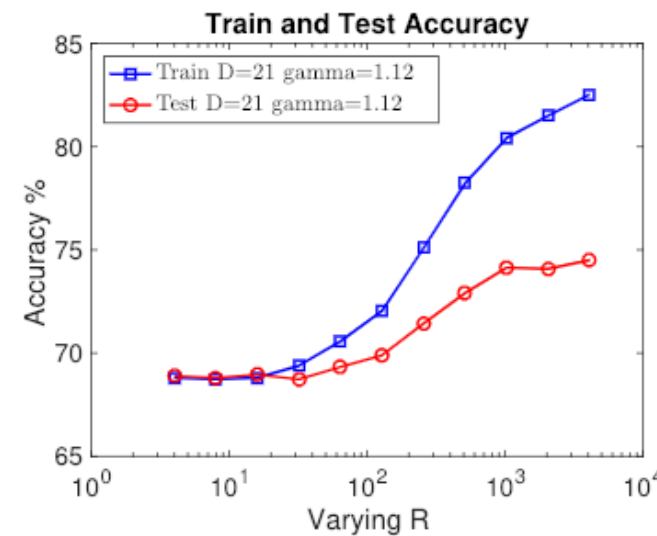




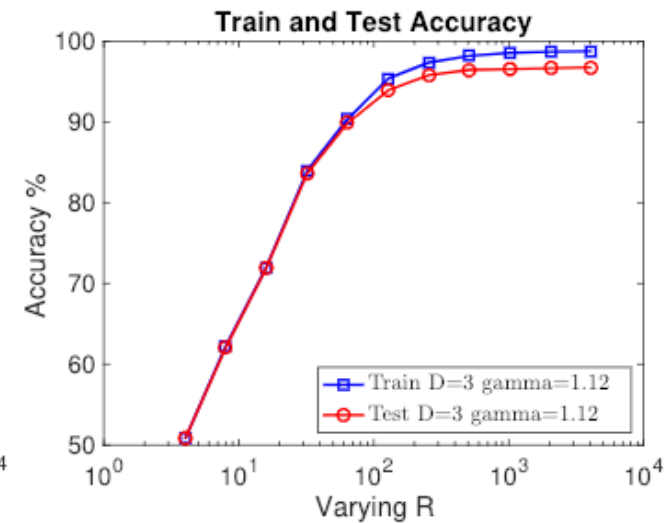
# Performance di WME variando i parametri

Variando R nel range [4, 4096]:

- All'aumentare di R, abbiamo un aumento di accuratezza



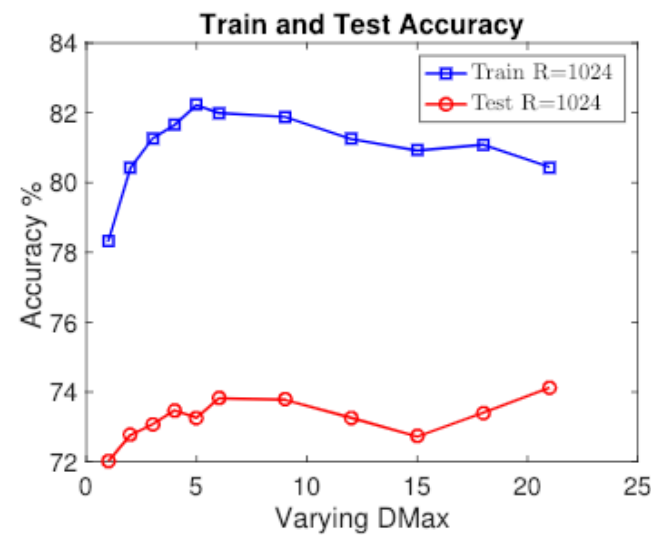
(a) TWITTER



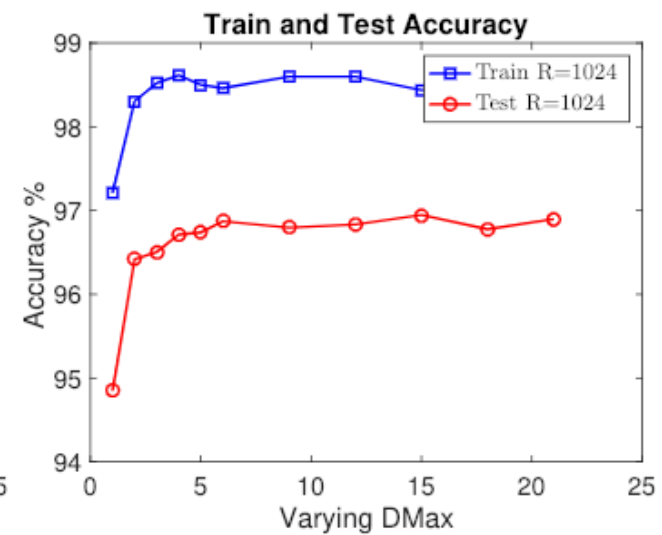
(b) CLASSIC

Impostando  $D_{min}=1$ , e variando  $D_{max}$  nel range [3, 21]:

- Il picco di accuratezza viene raggiunto con valori di D bassi ( $D \leq 6$ )



(a) TWITTER



(b) CLASSIC





# Confronto tra KNN + WMD e SVM + WME



L'esperimento consiste nel comparare la tecnica Word Mover's Distance accoppiato con il classificatore KNN, con la tecnica Word Mover's Embedding accoppiato con un modello di classificazione SVM

## Setup dell'esperimento:

- Dataset utilizzati: uguali all'esperimento precedente
- 10-cross validation per la scelta dei parametri ottimali
- Per ciascun metodo, due varianti: distanza fra tutte le parole del vocabolario della collezione precomputata (+P) o meno
- WME testato con valori alti e valori bassi di R [ WME(LR), WME(SR) ]





# Confronto tra KNN + WMD e SVM + WME

Classifier	KNN-WMD	KNN-WMD+P		WME(SR)	WME(SR)+P		WME(LR)	WME(LR)+P		
Dataset	Accu	Time	Time	Accu	Time	Time	Accu	Time	Time	Speedup
BBCSPORT	95.4 $\pm$ 1.2	147	122	95.5 $\pm$ 0.7	3	1	<b>98.2 <math>\pm</math> 0.6</b>	92	34	<b>122</b>
TWITTER	71.3 $\pm$ 0.6	25	4	72.5 $\pm$ 0.5	10	2	<b>74.5 <math>\pm</math> 0.5</b>	162	34	<b>2</b>
RECIPE	57.4 $\pm$ 0.3	448	326	57.4 $\pm$ 0.5	18	4	<b>61.8 <math>\pm</math> 0.8</b>	277	61	<b>82</b>
OHSUMED	55.5	3530	2807	55.8	24	7	<b>64.5</b>	757	240	<b>401</b>
CLASSIC	<b>97.2 <math>\pm</math> 0.1</b>	777	520	96.6 $\pm$ 0.2	49	10	97.1 $\pm$ 0.4	388	70	<b>52</b>
REUTERS	96.5	814	557	96.0	50	24	<b>97.2</b>	823	396	<b>23</b>
AMAZON	92.6 $\pm$ 0.3	2190	1319	92.7 $\pm$ 0.3	31	8	<b>94.3 <math>\pm</math> 0.4</b>	495	123	<b>165</b>
20NEWS	73.2	37988	32610	72.9	205	69	<b>78.3</b>	1620	547	<b>472</b>
RECIPE_L	71.4 $\pm$ 0.5	5942	2060	72.5 $\pm$ 0.4	113	20	<b>79.2 <math>\pm</math> 0.3</b>	1838	330	<b>103</b>

## Risultati:

- Accuratezza di WME(SR) e WMD comparabili. Complessità ridotta!
- WME(LR) ottiene valori di accuratezza maggiori
- Precomputazione delle distanze fra parole -> 3-5x speedup

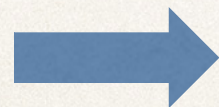




# Confronto tra WME e diversi metodi di Document Representation

## Comparazione di WME con 6 metodi di Document Representation

1. Smooth Inverse Frequency (SIF)
2. Word2Vec + NBOW
3. Word2Vec + TF\*IDF
4. PV-DBOW
5. PV-DM
6. Doc2VecC



Linear SVM per tutti i metodi





# Confronto tra WME e diversi metodi di Document Representation

Dataset	SIF(GloVe)	Word2Vec+nbow	Word2Vec+tf-idf	PV-DBOW	PV-DM	Doc2VecC	WME
BBCSPORT	97.3 $\pm$ 1.2	97.3 $\pm$ 0.9	96.9 $\pm$ 1.1	97.2 $\pm$ 0.7	97.9 $\pm$ 1.3	90.5 $\pm$ 1.7	<b>98.2 <math>\pm</math> 0.6</b>
TWITTER	57.8 $\pm$ 2.5	72.0 $\pm$ 1.5	71.9 $\pm$ 0.7	67.8 $\pm$ 0.4	67.3 $\pm$ 0.3	71.0 $\pm$ 0.4	<b>74.5 <math>\pm</math> 0.5</b>
OHSUMED	<b>67.1</b>	63.0	60.6	55.9	59.8	63.4	64.5
CLASSIC	92.7 $\pm$ 0.9	95.2 $\pm$ 0.4	93.9 $\pm$ 0.4	97.0 $\pm$ 0.3	96.5 $\pm$ 0.7	96.6 $\pm$ 0.4	<b>97.1 <math>\pm</math> 0.4</b>
REUTERS	87.6	96.9	95.9	96.3	94.9	96.5	<b>97.2</b>
AMAZON	94.1 $\pm$ 0.2	94.0 $\pm$ 0.5	92.2 $\pm$ 0.4	89.2 $\pm$ 0.3	88.6 $\pm$ 0.4	91.2 $\pm$ 0.5	<b>94.3 <math>\pm</math> 0.4</b>
20NEWS	72.3	71.7	70.2	71.0	74.0	78.2	<b>78.3</b>
RECIPE_L	71.1 $\pm$ 0.5	74.9 $\pm$ 0.5	73.1 $\pm$ 0.6	73.1 $\pm$ 0.5	71.1 $\pm$ 0.4	76.1 $\pm$ 0.4	<b>79.2 <math>\pm</math> 0.3</b>

WME performa consistentemente meglio rispetto agli altri metodi sulle collezioni testate





University of Padova

Department of Information Engineering

DEPARTMENT OF  
INFORMATION  
ENGINEERING  
UNIVERSITY OF PADOVA



# Implementazione di WME in Python e risultati sperimentali