Docente: Alessandro Mazzei

Studente: Michele Metta

La magia NER nascosta

L'obiettivo di questo progetto è stato quello di valutare le prestazioni per il task NER (Named Entity Recognition) di due modelli statistici:

- Hidden Markov Model (HMM)
- Maximum Entropy Markov Models (MEMM)

E di due baseline:

1) E' quella che data una certa parola, se questa è presente nel training set allora assegna a quella parola il tag più frequente che essa assume lì dentro, altrimenti qualora la parola fosse sconosciuta assegna ad essa il tag O.
2) E' quella che data una certa parola, se questa è presente nel training set allora assegna a quella parola il tag più frequente che essa assume lì dentro, altrimenti qualora la parola fosse sconosciuta assegna ad essa il tag B-MISC.

Gli esperimenti sono stati eseguiti su due corpus, uno di lingua italiana e l'altro di lingua inglese.

Sia per l'italiano che per l'inglese ci sono 3 diversi datasets:

1) Training set
2) Validation set
3) Test set

Le entità prese in considerazione in questi corpus sono le seguenti:

- PER (person)
- ORG (organization)
- LOC (location)
- MISC (miscellanea)

All'interno di ogni corpus delle due lingue ci sono delle frasi che sono state prese direttamente da Wikipedia, ad ogni parola di ogni frase è assegnato un certo tag secondo la tecnica BIO (Beginning-Inside-Outside). Il numero di tags finali è 9 e sono i seguenti:

- B-PER
- I-PER
- B-ORG
- I-ORG
- B-LOC
- I-LOC
- B-MISC
- I-MISC
- O

Tutto il codice è stato scritto utilizzando il linguaggio di programmazione Python.

Learning HMM

Per quanto riguarda la fase di learning dell'HMM per calcolare le probabilità di transizione da un certo tag ad un altro è stata utilizzata questa formula:

$$P(t_i|t_{i-1}) = \frac{\mathrm{C}(t_{i-1}, t_i)}{C(t_{i-1})}$$

Ove:

t_i: tag assegnato alla parola corrente.

t_{i-1}: tag assegnato alla parola precedente.

$\mathrm{C}(t_{i-1}, t_i)$: numero di volte che il tag t_i compare subito dopo il tag t_{i-1} nel training set.

$C(t_{i-1})$: numero di volte che il tag t_{i-1} compare nel training set nel training set.

A tutte le probabilità di transizione pari a 0 è stato assegnato un valore molto piccolo pari a:

$$\mathbf{10^{-80}}$$

questo è stato fatto ai fini di evitare eventuali errori su alcune frasi di test per le quali altrimenti la probabilità finale restituita dall'algoritmo di Viterbi sarebbe stata 0 su qualsiasi sequenza di tags.

Le probabilità di emissione invece sono state calcolate seguendo questa formula:

$$P(w_i|t_i) = \frac{\mathrm{C}(t_i, w_i)}{C(t_i)}$$

Ove:

t_i: tag assegnato alla parola corrente.

w_i: parola corrente.

$\mathrm{C}(t_i, w_i)$: numero di volte che tag t_i è il tag assegnato alla parola w_i nel training set.

$C(t_i)$: numero di volte che il tag t_i compare nel training set.

Sia le probabilità di transizione e sia le probabilità di emissione sono state pre-calcolate. Questo è stato fatto ai fini di velocizzare il più possibile la fase di learning dell'HMM in modo da poter predire tutti i tags di tutte le frasi presenti nei due test sets delle due lingue, permettendo di passare da circa 11 ore di esecuzione a pochi minuti. Sia le probabilità di transizione che di emissione per ogni corpus di ciascuna lingua sono stati memorizzati all'interno di due dizionari distinti che sono stati serializzati. L'algoritmo di Viterbi è stato l'algoritmo di programmazione dinamica che è stato implementato per la fase di decoding dell'HMM. Inoltre è stato applicato il logaritmo alle probabilità in modo tale da poter approssimare meglio i valori molto piccoli. Le strutture dati utilizzate nell'algoritmo di Viterbi sono le seguenti:

- **Matrice di Viterbi -> implementata con una matrice di dimensione (N+2, T).**

- **Backpointer -> implementato con un'altra matrice anch'essa di dimensione (N+2, T). Dove:**
  N: numero di tags totali (+2 perché si considera lo Start e l'End)
  T: numero di parole della frase di input.

Tecniche di Smoothing

Le tecniche di smoothing implementate per gestire le probabilità di emissione delle parole sconosciute nell'HMM sono le seguenti:

1) P(unk|O) = 1
2) P(unk|O) = P(unk|B-MISC) = 0.5
3) P(unk|t_i) = 1/9 (ovvero assegno la stessa probabilità ad ogni tags).
4) E' stata eseguita una statistica sui tags che vengono assegnati a tutte le parole che compaiono una sola volta nel validation set.

Valutazione

Per valutare i diversi algoritmi sono state implementate queste metriche:

- Accuracy generale (come per il PoS Tagging).
- Precision.
- Recall.
- Accuracy per ogni entity.

Accuratezza generale:

$$\boldsymbol{accuracy\ generale = \frac{numero\ tags\ predetti\ correttamente}{numero\ totale\ di\ tags\ presenti\ nel\ test\ set}}$$

Prima di calcolare la Precision e la Recall è stata eseguita una sorta di pulizia delle predizioni fatte dal modello in modo tale da sostituire tutte le predizioni non corrette dal punto di vista della tecnica BIO come ad esempio il fatto di predire per due parole direttamente i tags I-PER I-PER senza posizionare prima un B-PER. Quindi nel caso dell'esempio appena fatto i due tags I-PER I-PER sono stati sostituiti dai due tags O O, in questo modo è diventato più semplice riuscire a calcolare la Precision e la Recall.

La Precision è stata calcolata seguendo questa formula:

$$\boldsymbol{P = \frac{TP}{TP + FP}}$$

Ove:

TP (True Positives): è il numero di entità riconosciute completamente dal modello.

FP (False Positives): è il numero di volte che il modello ha predetto una certa entity (ad es: PER) invece di predire l'entity giusta (ad es: LOC), esempio:

B-PER I-PER invece di B-LOC I-LOC.

La Recall è stata calcolata seguendo questa formula:

$$R = \frac{TP}{numero\ totale\ di\ istanze\ di\ una\ certa\ entity\ nel\ test\ set}$$

L'accuracy per ogni entity è stata calcolata seguendo questi passi (in questo esempio consideriamo solo l'entità PER):

1) Calcolo del numero totale di B-PER e I-PER realmente presenti nel test set.
2) Calcolo del numero totale di B-Per e I-PER predetti correttamente dal modello.

Infine, il valore finale dell'accuratezza è stato calcolato facendo questo rapporto:

$$\frac{Numero\ totale\ di\ B-PER\ e\ I-PER\ predetti\ correttamente\ dal\ modello}{Numero\ totale\ di\ B-PER\ e\ I-PER\ realmente\ presenti\ nel\ test\ set}$$

Risultati ottenuti sul corpus in italiano

Tutti gli script in Python che sono stati utilizzati per eseguire gli esperimenti sul corpus in italiano sono presenti nella cartella chiamata "Test sul corpus in italiano".

Risultati baseline (parola sconosciuta -> B-MISC)

Accuracy generale
0.94

	Precision
PER	0.92
LOC	0.87
ORG	0.81
MISC	0.09

	Recall
PER	0.60
LOC	0.72
ORG	0.62
MISC	0.37

	Accuracy per ogni entity
PER	0.74
LOC	0.70
ORG	0.62
MISC	0.42
O	0.98

Il nome dello script in Python che è stato utilizzato per eseguire la baseline corrente e ottenere i risultati mostrati sopra è chiamato **"baseline_B-MISC_it.py"**.

Risultati baseline (parola sconosciuta -> O)

Accuracy generale
0.96

	Precision
PER	0.92
LOC	0.87
ORG	0.81
MISC	0.72

	Recall
PER	0.60
LOC	0.72
ORG	0.62
MISC	0.27

	Accuracy per ogni entity
PER	0.74
LOC	0.70
ORG	0.62
MISC	0.37
O	1.0

Il nome script in Python utilizzato per eseguire la baseline suddetta è **"baseline_O_it.py"**.

Commento: come si può notare la seconda baseline ottiene un'accuratezza generale maggiore della seconda, questo perché nel test set chiaramente alla maggior parte delle parole è assegnato il tag O e quindi è più probabile che la seconda baseline riesca ad etichettare correttamente una certa parola rispetto alla prima. Per quanto riguarda la Precision e la Recall esse sono abbastanza simili tranne sul fatto che la prima baseline ha una precisione molto bassa su MISC in quanto essa assegna alle parole sconosciute direttamente il tag B-MISC e quindi è molto facile che commetta un errore di falso positivo. La prima baseline però raggiunge un valore di Recall per l'entity MISC più elevata rispetto a quella raggiunta dalla seconda baseline e questo è comunque sempre legato al fatto che la prima assegna B-MISC alle parole sconosciute e quindi ha più probabilità di riuscire ad individuare un'entità MISC nel test set.

Risultati HMM (tecnica smoothing: P(unk|O) = 1)

Accuracy generale
0.967

	Precision
PER	0.96
LOC	0.88
ORG	0.82
MISC	0.77

	Recall
PER	0.69
LOC	0.76
ORG	0.72
MISC	0.37

	Accuracy per ogni entity
PER	0.80
LOC	0.77
ORG	0.73
MISC	0.51
O	1.0

Il nome dello script in Python utilizzato per eseguire l'HMM suddetto è **"Algoritmo di Viterbi (Tecnica sempre O).py"**.

Commento: osservando la tabella dei risultati dell'HMM con la prima tecnica di smoothing si capisce che quest'ultimo modello riesce a superare decisamente in termini di accuratezza generale, precision, recall e accuratezza di categoria anche la baseline migliore (ovvero quella che assegna alle parole sconosciute direttamente il tag O) su praticamente tutte le entità.

Risultati HMM

(tecnica smoothing: P(unk|O) = P(unk|B-MISC) = 0.5)

Accuracy generale
0.967

	Precision
PER	0.96
LOC	0.88
ORG	0.82
MISC	0.77

	Recall
PER	0.69
LOC	0.76
ORG	0.72
MISC	0.37

	Accuracy per ogni entity
PER	0.80
LOC	0.77
ORG	0.73
MISC	0.51
O	1.0

Il nome dello script in Python utilizzato per eseguire l'HMM suddetto è **"Algoritmo di Viterbi (Tecnica stessa distrib di prob su O e B-MISC).py"**.

Risultati HMM

(tecnica smoothing: P(unk|t_i) = 1/9)

Accuracy generale
0.97

	Precision
PER	0.96
LOC	0.88
ORG	0.82
MISC	0.79

	Recall
PER	0.79
LOC	0.76
ORG	0.72
MISC	0.38

	Accuracy per ogni entity
PER	0.86
LOC	0.77
ORG	0.74
MISC	0.52
O	1.0

Il nome dello script in Python utilizzato per eseguire l'HMM suddetto è **"Algoritmo di Viterbi (Tecnica stessa distrib di prob su tutti i tag).py"**.

Risultati HMM

(tecnica smoothing: statistica sul validation set)

Accuracy generale
0.967

	Accuracy per ogni entity
PER	0.80
LOC	0.77
ORG	0.73
MISC	0.51
O	1.0

	Precision
PER	0.95
LOC	0.88
ORG	0.82
MISC	0.77

	Recall
PER	0.69
LOC	0.76
ORG	0.72
MISC	0.37

Il nome dello script in Python utilizzato per eseguire l'HMM suddetto è **"Algoritmo di Viterbi (Tecnica che utilizza il development set).py"**.

Commento: dai risultati ottenuti si può notare come l'HMM con qualsiasi tecnica di smoothing si comporta in generale nettamente meglio un pò con tutte le metriche rispetto ad una qualsiasi delle due baselines. **In particolare, l'HMM con la terza tecnica di smoothing risulta essere il migliore per il NER sulla lingua italiana sia per quanto riguarda l'accuratezza generale che per la precision, recall e accuracy per ogni entity.**

Risultati MEMM

Features utilizzate per ogni parola di una frase:

1) Parola corrente.
2) Tag assegnato alla parola precedente.
3) Il prefisso della parola corrente (prime 3 lettere della parola).
4) Il suffisso della parola corrente (ultime 2 lettere della parola).
5) Feature booleana: 1 se la stringa corrente è un numero, altrimenti 0.
6) Feature booleana: 1 se la prima lettera della parola corrente è in maiuscolo, altrimenti 0.
7) Se la parola corrente non è la prima parola della frase prende come unica feature sia la parola corrente che quella precedente.
8) Feature booleana: 1 se la parola corrente è la prima parola della frase, altrimenti 0.
9) Se la parola corrente non è l'ultima parola della frase prende come unica feature sia la parola corrente che quella successiva.
10) Feature booleana: 1 se la parola corrente è l'ultima parola della frase, altrimenti 0.
11) Se la parola corrente non è la prima parola e neanche l'ultima della frase allora prende come unica feature: la parola corrente, la parola precedente e la parola successiva.

Accuracy generale
0.98

	Precision
PER	0.87
LOC	0.81
ORG	0.87
MISC	0.77

	Recall
PER	0.86
LOC	0.89
ORG	0.73
MISC	0.59

	Accuracy per ogni entity
PER	0.90
LOC	0.89
ORG	0.74
MISC	0.70
O	1.0

Il nome dello script in Python utilizzato per eseguire il MEMM sull'italiano è presente nella cartella chiamata MEMM ed è chiamato **"memm_tagger_it.py"**.

Commento: Dai risultati ottenuti per il MEMM si può notare come quest'ultimo sull'italiano risulta essere il modello migliore dal punto di vista dell'accuracy generale, per quanto riguarda invece la Precision il modello migliore sulle entità PERSON, LOCATION e MISC continua ad essere l'HMM con la terza tecnica di smoothing in quanto il valore di tale metrica risulta essere superiore. Questo sta ad indicare che quando questo modello predice una sequenza di parole come un'unica entità tra quelle tre specificate pocanzi è molto più probabile che questa predizione sia corretta. Invece per quanto riguarda la Recall, il MEMM è il modello migliore su tutte le entità, questo indica sostanzialmente che questo modello è nettamente superiore dal punto di vista della completezza in quanto valori di Recall più elevanti indicano che il modello è in grado di riuscire a trovare un numero maggiore di istanze presenti nel test set per una certa entity. Infine, anche dal punto di vista dell'accuratezza per ogni entity, il MEMM supera decisamente l'HMM migliore.

Infine, è possibile notare come i vari HMMs con le varie tecniche di smoothing e il MEMM continuino a trovare difficoltà nel riuscire a riconoscere l'entità MISC, in quanto sia i valori della Precision, della Recall e dell'accuracy per ogni entity, risultano essere quasi sempre più bassi rispetto a quelli per le altre entità.

Risultati ottenuti sul corpus in inglese

Tutti gli script in Python che sono stati utilizzati per eseguire gli esperimenti sul corpus in inglese sono presenti nella cartella chiamata "Test sul corpus in inglese".

Risultati baseline (parola sconosciuta -> B-MISC)

Accuracy generale
0.93

	Precision
PER	0.81
LOC	0.73
ORG	0.70
MISC	0.23

	Recall
PER	0.46
LOC	0.62
ORG	0.43
MISC	0.51

	Accuracy per ogni entity
PER	0.63
LOC	0.66
ORG	0.53
MISC	0.50
O	0.98

Il nome dello script in Python che è stato utilizzato per eseguire la baseline corrente e ottenere i risultati mostrati sopra è chiamato **"baseline_B-MISC_en.py"**.

Risultati baseline (parola sconosciuta -> O)

Accuracy generale
0.94

	Precision
PER	0.81
LOC	0.74
ORG	0.70
MISC	0.74

	Recall
PER	0.46
LOC	0.62
ORG	0.43
MISC	0.40

	Accuracy per ogni entity
PER	0.63
LOC	0.66
ORG	0.53
MISC	0.43
O	1.0

Il nome script in Python utilizzato per eseguire la baseline suddetta è **"baseline_O_en.py"**.

Risultati HMM (tecnica smoothing: P(unk|O) = 1)

Accuracy generale
0.952

	Precision
PER	0.88
LOC	0.75
ORG	0.77
MISC	0.72

	Recall
PER	0.54
LOC	0.69
ORG	0.50
MISC	0.43

	Accuracy per ogni entity
PER	0.68
LOC	0.71
ORG	0.59
MISC	0.52
O	1.0

Il nome dello script in Python utilizzato per eseguire l'HMM suddetto è **"Algoritmo di Viterbi (Tecnica sempre O).py"**.

Commento: Anche in questo caso come nell'italiano l'HMM con la tecnica di smoothing più semplice (ovvero la prima) riesce a superare decisamente in termini di accuratezza generale, precision, recall e accuratezza di categoria anche la baseline migliore (quella che assegna alle parole sconosciute direttamente il tag O) su praticamente tutte le entità.

Risultati HMM

(tecnica smoothing: P(unk|O) = P(unk|B-MISC) = 0.5)

Accuracy generale
0.952

	Precision
PER	0.88
LOC	0.75
ORG	0.77
MISC	0.72

	Recall
PER	0.54
LOC	0.69
ORG	0.50
MISC	0.43

	Accuracy per ogni entity
PER	0.68
LOC	0.71
ORG	0.59
MISC	0.52
O	1.0

Il nome dello script in Python utilizzato per eseguire l'HMM suddetto è **"Algoritmo di Viterbi (Tecnica stessa distrib di prob su O e B-MISC).py"**.

Risultati HMM

(tecnica smoothing: P(unk|t_i) = 1/9)

Accuracy generale
0.956

	Precision
PER	0.90
LOC	0.75
ORG	0.78
MISC	0.72

	Recall
PER	0.68
LOC	0.69
ORG	0.50
MISC	0.43

	Accuracy per ogni entity
PER	0.77
LOC	0.72
ORG	0.60
MISC	0.52
O	1.0

Il nome dello script in Python utilizzato per eseguire l'HMM suddetto è **"Algoritmo di Viterbi (Tecnica stessa distrib di prob su tutti i tag).py"**.

Risultati HMM

(tecnica smoothing: statistica sul validation set)

Accuracy generale
0.952

	Precision
PER	0.88
LOC	0.75
ORG	0.77
MISC	0.72

	Recall
PER	0.54
LOC	0.69
ORG	0.50
MISC	0.43

	Accuracy per ogni entity
PER	0.68
LOC	0.72
ORG	0.59
MISC	0.52
O	1.0

Il nome dello script in Python utilizzato per eseguire l'HMM suddetto è **"Algoritmo di Viterbi (Tecnica che utilizza il development set).py"**.

Commento: Dai risultati ottenuti si può notare come in generale le performance (considerando le varie metriche) sia delle baselines che dei vari modelli sulla lingua inglese risultano essere più basse rispetto a quelle ottenute sulla lingua italiana. Il miglior modello tra quelli visti fino ad ora risulta essere l'HMM con la terza tecnica di smoothing proprio come nell'italiano. Le differenze si notano soprattutto per quanto riguarda l'accuracy generale che viene incrementata dell'1% con la terza tecnica, nella Precision per l'entità PERSON, in cui l'HMM migliore raggiunge il 90% e nella Recall sempre dell'entità PERSON in cui sempre l'HMM con la terza tecnica di smoothing ha un incremento circa del 14% rispetto alle altre tecniche. Inoltre anche l'accuracy per ogni entity viene

incrementata dall'HMM migliore sia per PERSON che per ORGANIZATION, rispettivamente dell'9% e dell'1%.

Risultati MEMM

Le features utilizzate per il MEMM addestrato sul training set della lingua inglese sono identiche a quelle utilizzate per quello addestrato sul training set della lingua italiana.

Accuracy generale
0.964

	Precision
PER	0.78
LOC	0.66
ORG	0.83
MISC	0.75

	Recall
PER	0.79
LOC	0.83
ORG	0.62
MISC	0.58

	Accuracy per ogni entity
PER	0.82
LOC	0.83
ORG	0.61
MISC	0.66
O	1.0

Il nome dello script in Python utilizzato per eseguire il MEMM sull'inglese è presente sempre nella cartella chiamata MEMM ed è chiamato **"memm_tagger_en.py"**.

Commento: Dai risultati ottenuti per il MEMM si può notare come quest'ultimo anche sull'inglese risulta essere il modello migliore dal punto di vista dell'accuracy generale, per quanto riguarda invece la Precision sull'inglese il modello migliore sulle entità PERSON e LOCATION continua ad essere l'HMM con la terza tecnica di smoothing in quanto il valore di tale metrica risulta essere superiore. Invece per quanto riguarda la Recall e l'accuracy per ogni entity, il MEMM anche per la lingua inglese risulta essere il modello migliore su tutte le entità.

Infine, anche sulla lingua inglese è possibile notare come i vari HMMs con le varie tecniche di smoothing e il MEMM continuino a trovare difficoltà nel riuscire a riconoscere l'entità MISC, in quanto anche in questo caso i valori della Precision, Recall e dell'accuracy per ogni entity, risultano essere quasi sempre più bassi rispetto a quelli per le altre entità.

TEST FRASI FUORI DOMINIO

Ogni modello nominato precedentemente e ogni baseline citata sono stati testati su 3 frasi fuori dominio in lingua italiana e 3 frasi in lingua inglese. Ovviamente i modelli addestrati sul corpus in italiano sono stati testati sulle frasi in italiano e quelli addestrati su quello inglese sono stati testati su quelli in inglese, stessa cosa vale per le 2 baselines. Ad ogni parola di ogni frase è stato assegnato un certo tag reale.

RISULTATI SU FRASI FUORI DOMINIO (italiano)

Le frasi sono le seguenti:

```
0   La  O
1   vera    O
2   casa    O
3   di  O
4   Harry   B-PER
5   Potter  I-PER
6   è   O
7   il  O
8   castello    B-LOC
9   di  I-LOC
10  Hogwards    I-LOC
11  .   O

0   Harry   B-PER
1   le  O
2   raccontò    O
3   del O
4   loro    O
5   incontro    O
6   a   O
7   Diagon  B-LOC
8   Alley   I-LOC
9   .   O

0   Mr.  B-PER
1   Dursley I-PER
2   era O
3   direttore   O
4   di  O
5   una O
6   ditta   O
7   di  O
8   nome    O
9   Grunnings  B-ORG
10  ,   O
11  che O
12  frabbricava O
13  trapani O
14  .   O
```

PREDIZIONI E RISULTATI OTTENUTI SULLE FRASI FUORI DOMINIO

Baseline (parole sconosciute -> B-MISC):

	Tag predetto
La	O
Vera	O
casa	O
di	O
Harry	B-PER
Potter	I-MISC
è	O
Il	O
castello	O
di	O
Hogwards	B-MISC
.	O

	Tag predetto
Harry	B-PER
Le	O
Raccontò	O
del	O
loro	O
Incontro	O
a	O
Diagon	B-MISC
Alley	I-PER
.	O

	Tag predetto
Mr.	B-MISC
Dursley	B-MISC
era	O
direttore	O
di	O
una	O
ditta	O
di	O
nome	O
Grunnings	B-MISC
,	O
che	O
fabbricava	B-MISC
trapani	B-MISC
.	O

Baseline (parole sconosciute -> O):

	Tag predetto
La	O
Vera	O
casa	O
di	O
Harry	B-PER
Potter	I-MISC
è	O
Il	O
castello	O
di	O
Hogwards	O
.	O

	Tag predetto
Harry	B-PER
Le	O
Raccontò	O
del	O
loro	O
Incontro	O
a	O
Diagon	O
Alley	I-PER
.	O

	Tag predetto
Mr.	B-MISC
Dursley	O
era	O
direttore	O
di	O
una	O
ditta	O
di	O
nome	O
Grunnings	O
,	O
che	O
fabbricava	O
trapani	O
.	O

Il nome dello script in Python utilizzato per testare le due baselines sulle frasi fuori dominio è presente sempre all'interno della cartella chiamata "Test sul corpus in italiano" ed è chiamato **"Test sulle frasi fuori dominio baselines.py"**.

HMM (prima tecnica di smoothing):

	Tag predetto
La	O
Vera	O
casa	O
di	O
Harry	B-MISC
Potter	I-MISC
è	O
Il	O
castello	O
di	O
Hogwards	O
.	O

	Tag predetto
Harry	B-PER
Le	O
Raccontò	O
del	O
loro	O
Incontro	O
a	O
Diagon	O
Alley	I-PER
.	O

	Tag predetto
Mr.	B-MISC
Dursley	O
era	O
direttore	O
di	O
una	O
ditta	O
di	O
nome	O
Grunnings	O
,	O
che	O
fabbricava	O
trapani	O
.	O

HMM (seconda tecnica di smoothing):

	Tag predetto
La	O
Vera	O
casa	O
di	O
Harry	B-MISC
Potter	I-MISC
è	O
Il	O
castello	O
di	O
Hogwards	O
.	O

	Tag predetto
Harry	B-PER
Le	O
Raccontò	O
del	O
loro	O
Incontro	O
a	O
Diagon	O
Alley	I-PER
.	O

	Tag predetto
Mr.	B-MISC
Dursley	O
era	O
direttore	O
di	O
una	O
ditta	O
di	O
nome	O
Grunnings	O
,	O
che	O
fabbricava	O
trapani	O
.	O

HMM (terza tecnica di smoothing):

	Tag predetto
La	O
Vera	O
casa	O
di	O
Harry	B-MISC
Potter	I-MISC
è	O
Il	O
castello	O
di	O
Hogwards	O
.	O

	Tag predetto
Harry	B-PER
Le	O
Raccontò	O
del	O
loro	O
Incontro	O
a	O
Diagon	O
Alley	I-PER
.	O

	Tag predetto
Mr.	B-MISC
Dursley	I-MISC
era	O
direttore	O
di	O
una	O
ditta	O
di	O
nome	O
Grunnings	O
,	O
che	O
fabbricava	O
trapani	O
.	O

HMM (quarta tecnica di smoothing):

	Tag predetto
La	O
Vera	O
casa	O
di	O
Harry	B-MISC
Potter	I-MISC
è	O
Il	O
castello	O
di	O
Hogwards	O
.	O

	Tag predetto
Harry	B-PER
Le	O
Raccontò	O
del	O
loro	O
Incontro	O
a	O
Diagon	O
Alley	I-PER
.	O

	Tag predetto
Mr.	B-MISC
Dursley	O
era	O
direttore	O
di	O
una	O
ditta	O
di	O
nome	O
Grunnings	O
,	O
che	O
fabbricava	O
trapani	O
.	O

Il nome dello script in Python utilizzato per testare i vari modelli HMMs con le varie tecniche sulle frasi fuori dominio è presente sempre all'interno della cartella chiamata "Test sul corpus in italiano" ed è chiamato **"Test sulle frasi fuori dominio.py"**. Ogni volta che bisognava testare una

tecnica di smoothing diversa questa veniva settata all'interno della funzione chiamata "calcolo_prob_di_emissione" presente sempre all'interno dello script già citato.

MEMM:

	Tag predetto
La	O
Vera	O
casa	O
di	O
Harry	B-MISC
Potter	I-MISC
è	O
Il	O
castello	B-LOC
di	I-LOC
Hogwards	I-LOC
.	O

	Tag predetto
Harry	B-PER
Le	O
Raccontò	O
del	O
loro	O
Incontro	O
a	O
Diagon	B-PER
Alley	I-PER
.	O

	Tag predetto
Mr.	O
Dursley	B-PER
era	O
direttore	O
di	O
una	O
ditta	O
di	O
nome	O
Grunnings	B-MISC
,	O
che	O
fabbricava	O
trapani	O
.	O

Il nome dello script utilizzato per testare il MEMM sulle frasi fuori dominio in italiano è presente sempre nella cartella chiamata "MEMM" ed è chiamato **"Test_frasi_fuori_dominio_MEMM_it.py"**.

RISULTATI SU FRASI FUORI DOMINIO (inglese)

Le frasi sono le seguenti:

```
0   Harry   B-PER
1   Potter  I-PER
2   's  O
3   real    O
4   home    O
5   is  O
6   Hogwarts    B-LOC
7   castle  I-LOC
8   .   O

0   Harry   B-PER
1   told    O
2   her O
3   about   O
4   their   O
5   meeting O
6   in  O
7   Diagon  B-LOC
8   Alley   I-LOC
9   .   O

0   Mr. B-PER
1   Dursley I-PER
2   was O
3   the O
4   director    O
5   of  O
6   a   O
7   firm    O
8   called  O
9   Grunnings   B-ORG
10  ,   O
11  which   O
12  made    O
13  drills  O
14  .   O
```

PREDIZIONI E RISULTATI OTTENUTI SULLE FRASI FUORI DOMINIO

Baseline (parole sconosciute -> B-MISC):

	Tag predetto
Harry	B-PER
Potter	I-MISC
‘s	O
real	O
home	O
is	O
Hogwarts	B-MISC
castle	I-LOC
.	O

	Tag predetto
Harry	B-PER
told	O
her	O
about	O
their	O
meeting	O
in	O
Diagon	O
Alley	I-MISC
.	O

	Tag predetto
Mr.	B-MISC
Dursley	B-LOC
was	O
the	O
director	O
of	O
a	O
firm	O
called	O
Grunnings	O
,	O
which	O
made	O
drills	O
.	O

Baseline (parole sconosciute -> O):

	Tag predetto
Harry	B-PER
Potter	I-MISC
‘s	O
real	O
home	O
is	O
Hogwarts	B-MISC
castle	I-LOC
.	O

	Tag predetto
Harry	B-PER
told	O
her	O
about	O
their	O
meeting	O
in	O
Diagon	O
Alley	I-MISC
.	O

	Tag predetto
Mr.	B-MISC
Dursley	B-LOC
was	O
the	O
director	O
of	O
a	O
firm	O
called	O
Grunnings	O
,	O
which	O
made	O
drills	O
.	O

Il nome dello script in Python utilizzato per testare le due baselines sulle frasi fuori dominio è presente sempre all’interno della cartella chiamata “Test sul corpus in inglese” ed è chiamato **“Test sulle frasi fuori dominio baselines.py”**.

HMM (prima tecnica di smoothing):

	Tag predetto
Harry	B-PER
Potter	I-PER
‘s	O
real	O
home	O
is	O
Hogwarts	B-MISC
castle	I-MISC
.	O

	Tag predetto
Harry	B-PER
told	O
her	O
about	O
their	O
meeting	O
in	O
Diagon	O
Alley	B-LOC
.	O

	Tag predetto
Mr.	B-MISC
Dursley	B-LOC
was	O
the	O
director	O
of	O
a	O
firm	O
called	O
Grunnings	O
,	O
which	O
made	O
drills	O
.	O

HMM (seconda tecnica di smoothing):

	Tag predetto
Harry	B-PER
Potter	I-PER
‘s	O
real	O
home	O
is	O
Hogwarts	B-MISC
castle	I-MISC
.	O

	Tag predetto
Harry	B-PER
told	O
her	O
about	O
their	O
meeting	O
in	O
Diagon	O
Alley	B-LOC
.	O

	Tag predetto
Mr.	B-MISC
Dursley	B-LOC
was	O
the	O
director	O
of	O
a	O
firm	O
called	O
Grunnings	O
,	O
which	O
made	O
drills	O
.	O

HMM (terza tecnica di smoothing):

	Tag predetto
Harry	B-PER
Potter	I-PER
‘s	O
real	O
home	O
is	O
Hogwarts	B-MISC
castle	I-MISC
.	O

	Tag predetto
Harry	B-PER
told	O
her	O
about	O
their	O
meeting	O
in	O
Diagon	O
Alley	B-LOC
.	O

	Tag predetto
Mr.	B-MISC
Dursley	B-LOC
was	O
the	O
director	O
of	O
a	O
firm	O
called	O
Grunnings	O
,	O
which	O
made	O
drills	O
.	O

HMM (quarta tecnica di smoothing):

	Tag predetto
Harry	B-PER
Potter	I-PER
‘s	O
real	O
home	O
is	O
Hogwarts	B-MISC
castle	I-MISC
.	O

	Tag predetto
Harry	B-PER
told	O
her	O
about	O
their	O
meeting	O
in	O
Diagon	O
Alley	B-LOC
.	O

	Tag predetto
Mr.	B-MISC
Dursley	B-LOC
was	O
the	O
director	O
of	O
a	O
firm	O
called	O
Grunnings	O
,	O
which	O
made	O
drills	O
.	O

Il nome dello script in Python utilizzato per testare i vari modelli HMMs con le varie tecniche sulle frasi fuori dominio è presente sempre all’interno della cartella chiamata “Test sul corpus in inglese” ed è chiamato **“Test sulle frasi fuori dominio.py”**. Anche in questo per settare la tecnica di smoothing da testare si esegue lo stesso procedimento fatto per l’italiano.

MEMM:

	Tag predetto
Harry	B-PER
Potter	I-PER
‘s	O
real	O
home	O
is	O
Hogwarts	B-MISC
castle	I-MISC
.	O

	Tag predetto
Harry	B-PER
told	O
her	O
about	O
their	O
meeting	O
in	O
Diagon	B-LOC
Alley	I-LOC
.	O

	Tag predetto
Mr.	O
Dursley	B-PER
was	O
the	O
director	O
of	O
a	O
firm	O
called	O
Grunnings	B-MISC
,	O
which	O
made	O
drills	O
.	O

Il nome dello script utilizzato per testare il MEMM sulle frasi fuori dominio in italiano è presente sempre nella cartella chiamata “MEMM” ed è chiamato **“Test_frasi_fuori_dominio_MEMM_en.py”**.

Commento finale: Dalle varie tabelle contenti le predizioni fatte dai vari algoritmi sulle varie frasi fuori dominio si può notare come per quanto riguarda l’italiano le predizioni fatte dai vari modelli HMMs con le varie tecniche di smoothing risultato essere molto simili e non molto corrette, mentre per quanto riguarda il MEMM le cose vanno leggermente meglio ma comunque anche quest’ultimo modello trova difficoltà ad individuare correttamente le giuste entità presenti nelle varie frasi. Per quanto riguarda invece le predizioni fatte sulle frasi fuori dominio in lingua inglese anche in questo caso i vari modelli HMMs non sembrano riuscire ad individuare correttamente tutte le varie entità mentre per quanto riguarda il MEMM sicuramente è facile notare come quest’ultimo si comporti meglio rispetto ai modelli citati precedentemente.