



UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

TRIENNALE IN

METODI PER IL RITROVAMENTO DELLE INFORMAZIONI

RICONOSCIMENTO DEL BISOGNO INFORMATIVO DA RICHIESTE IN
LINGUAGGIO NATURALE IN AMBITO TURISTICO

Relatori:

Dott. Marco De Gemmis

Dott. Marco Polignano

Laureando:

Giuseppe Mazzone

ANNO ACCADEMICO 2018/2019

La chiave del successo è amare il proprio lavoro e viverlo come un gioco.

Sommario

CAPITOLO 1 – INTRODUZIONE	5
1.1 OBIETTIVO DELLA TESI	7
CAPITOLO 2 – STATO DELL’ARTE	9
2.1 PRE-ELABORAZIONE DEL TESTO	9
2.1.1 RIMOZIONE DELLE STOP WORDS	10
2.1.2 TOKENIZATION	11
2.1.3 LEMMATIZATION & STEMMING	11
2.1.4 POS TAGGING	11
2.2 WORD REPRESENTATION	12
2.2.1 BAG OF WORDS	12
2.2.2 TF-IDF	12
2.2.3 WORD EMBEDDING	14
2.3 INTENT DETECTION	19
2.3.1 CLUSTERING	20
2.3.2 SVM	20
2.3.3 CLASSIFICATORE BAYESIANO	21
2.3.4 CASE-BASED	22
2.3.5 ALBERI DECISIONALI	23
2.3.6 RETI NEURALI	24
CAPITOLO 3 - STATO DELLA PRATICA	36
3.1 API NLP	36
3.2 LIBRERIE PYTHON DISPONIBILI	43
3.2.1 GENSIM	43
3.2.2 NLTK	43
3.2.3 SPACY	43
3.2.4 SCIKIT-LEARN	44
3.2.5 TENSORFLOW	45
3.2.6 KERAS	46
3.2.7 PYTORCH	47
3.2.8 HYPEROPT	48
3.2.9 JOBLIB	49
CAPITOLO 4 - ARCHITETTURA E SVILUPPO	51
4.1 SCRAPPING DATASET	52
4.2 PRE-ELABORAZIONE DEL TESTO	55

4.3 WORD REPRESENTATION	55
4.4 INTENT DETECTION.....	57
CAPITOLO 5 – VALUTAZIONE.....	62
5.1 METRICHE DI VALUTAZIONE.....	63
5.2 TRAINING/TEST SPLIT VALIDATION	64
5.2 CROSS-VALIDATION	65
5.3 McNEMAR TEST	66
5.4 RISULTATI OTTENUTI	68
CAPITOLO 6 – CONCLUSIONI E SVILUPPI FUTURI	71
6.1 CONCLUSIONI	71
6.2 SVILUPPI FUTURI.....	73
RIFERIMENTI BIBLIOGRAFICI	76

CAPITOLO 1 – INTRODUZIONE

I moderni motori di ricerca vanno oltre il recupero di documenti pertinenti.

Per soddisfare le esigenze di informazioni di un utente, essi mirano anche a visualizzare una risposta concisa alla sua domanda. Pertanto, richiedono una conoscenza approfondita della query dell'utente. L'identificazione del bisogno informativo insito all'interno di una domanda o query risulta essere un passaggio cruciale verso il raggiungimento di una risposta concisa. Questo non solo perché aiuta a visualizzare risultati di ricerca arricchiti semanticamente, ma anche perché aiuta a migliorare i risultati di posizionamento attivando un motore di ricerca verticale in un determinato dominio (ad esempio, ricerca di immagini, ricerca di notizie e motori di ricerca di lavoro) fungendo da filtro su un insieme di risultati possibili.

Il rilevamento delle intenzioni contenute in una query o più comunemente chiamato, Query Intent Detection, risulta essere un'attività impegnativa poiché le query sono in genere brevi e, l'identificazione dell'esatto intento dell'utente richiede più contesto oltre le parole chiave per poter essere accurato. Inoltre, il numero di categorie di intenti potrebbe essere molto elevato. La maggior parte degli approcci richiede un'enorme quantità di sforzi umani per far fronte a queste sfide, sia definendo modelli per ciascuna classe di intenti o definendo caratteristiche discriminatorie per le query per eseguire modelli statistici. In questa tesi, vengono per prima cosa identificate quali sono le classi di intent possibili e subito dopo, utilizzando un dataset propriamente creato, vengono addestrati e poi valutati molti degli algoritmi adottati per il Query Intent Detection presenti allo stato dell'arte.

In questo documento, presentiamo il nostro approccio all'identificazione dell'intento di query come attività di classificazione multi-classe con rappresentazioni di vettore di query estratte come funzionalità.

Il nostro approccio è quello di utilizzare l'apprendimento profondo per trovare rappresentazioni di vettori di query, per addestrare successivamente un modello di predizione in grado di riconoscere l'intent di una nuova query posta in input.

Uno dei vantaggi dell'utilizzo delle rappresentazioni del vettore di query è l'incorporamento di query in uno spazio, ovvero l'embedding, in modo tale che query semanticamente simili siano vicine tra loro. Ad esempio, le domande "Dove posso cenare stasera?" e "Cosa posso mangiare stasera?" saranno vicine l'una all'altra; pertanto, possono essere assegnati lo stesso intento.

L'apprendimento profondo viene utilizzato principalmente nelle attività di elaborazione del testo sfruttando le rappresentazioni vettoriali di parole (come word2vec che sono essenzialmente vettori con caratteristiche semantiche codificate nelle loro dimensioni. In questo documento apprendiamo rappresentazioni di vettore di query che utilizzano reti neurali convoluzionali (CNN) che sono addestrate su rappresentazioni di vettore di parole. La CNN è stata originariamente inventata per la visione artificiale e recentemente ha dimostrato di essere utile in molte attività di elaborazione del linguaggio naturale e di recupero delle informazioni, come l'analisi semantica, la modellizzazione delle frasi, la classifica dei documenti, documentare la somiglianza e riformulare le query.

I risultati sperimentali mostrano che il modello di rete neurale C-BiLSTM proposto, corrispondente ad una rete CNN seguita da una LSTM Bi-direzionale, è in grado di rilevare efficacemente gli intenti delle query con maggiore accuratezza rispetto agli altri metodi testati.

1.1 OBIETTIVO DELLA TESI

La Puglia è conosciuta in tutta Europa per la calda accoglienza che la gente del posto riesce a fornire all'ospite, non a caso il detto popolare "A baar nsciun ie frastiir" (A Bari nessuno è straniero) è molto popolare a Bari e in tutta la Puglia.

Tale detto può essere considerato la sintesi estrema del progetto "Feel at Home".

Lo scopo del progetto è supportare l'ospite favorendone il contatto, l'esperienza, l'incontro fra i locali (residenti), comunità locali e piccoli operatori economici (POE), facendo sentire l'ospite come a casa.

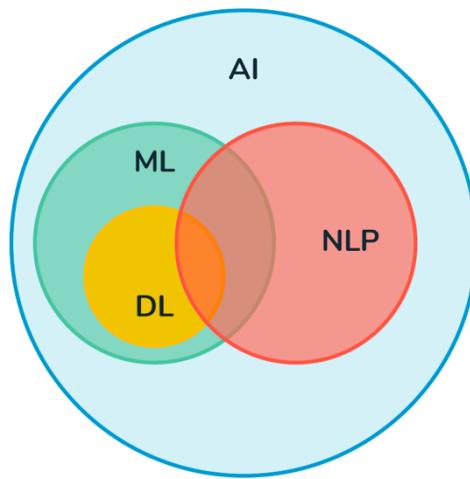
È infatti di interesse riuscire a realizzare un ambiente di semplice utilizzo che permetta la nascita e la crescita di rapporti di fiducia fra coloro che arrivano in un posto nuovo senza conoscerne le peculiarità e coloro che possono aiutarli, anche in situazioni di emergenza, con suggerimenti e/o servizi.

Quindi sarà necessario progettare un sistema intelligente con cui l'ospite potrà interagire effettuando una query esplicita in linguaggio naturale, che venga elaborata dal sistema, in modo tale da proporre un risultato all'ospite attraverso una lista di soggetti che potrebbero soddisfare le sue necessità e con i quali potrà decidere di mettersi in contatto. Tale suggerimento sarà proposto come forma di interazione veloce ed immediata per cercare di prevedere cosa potrebbe essere utile per l'ospite.

È importante notare che per far svolgere questo compito ad un umano è necessaria molta conoscenza da parte sua; una macchina creata nello specifico, richiede meno conoscenza e diventa sempre più esperto con il tempo (Knowledge-intensive)

Come già discusso nell'introduzione, un buon punto di partenza per rispondere alla query posta dall'utente, sarebbe quella di definire il bisogno informativo insito all'interno. La classificazione / estrazione attenta di tale intento, sembra essere molto più appropriata in un ambiente di controllo qualità in cui è necessario estrarre l'intento da una domanda o da una query ad hoc. In questo caso, possiamo risolvere il problema quasi esclusivamente in modo supervisionato, cioè, definendo un insieme di intenti predefiniti e imparando un classificatore dandogli come dati di allenamento delle query già pre-categorizzate.

L'obiettivo di questa tesi è quello di creare un servizio in grado di riconoscere gli intent all'interno di una query posta dall'utente in modo tale da utilizzarlo come filtro in un motore di ricerca verticale che sarà così in grado di filtrare la lista di soggetti di output richiesta dal progetto in base al bisogno informativo della query.



Tale obiettivo comporta un ampio studio nell'ambito dell'NLP e del Machine Learning e più in generale dell'Intelligenza Artificiale. Nel prossimo capitolo verrà fornita una panoramica generale sulla letteratura recentemente prodotta nei vari ambiti di ricerca che coinvolgono (o che possono coinvolgere) l'intero processo di analisi della query dell'utente, partendo dalle tecniche di rappresentazione del linguaggio naturale, passando per tecniche/algoritmi di comprensione dell'intent (intent detection).

CAPITOLO 2 – STATO DELL’ARTE

In questo capitolo verranno illustrate le principali tecniche di intent Detection a partire da testo presenti a stato dell’arte, includendo anche le varie tecniche di rappresentazione del testo.

La maggior parte delle informazioni attualmente disponibili sono fruibili in formato digitale, e sono organizzate in forma non strutturata, ossia sottoforma di testi o frammenti di testo.

Dal punto di vista della rappresentazione della conoscenza, il testo è un formato molto diverso rispetto a quello strutturato con cui sono memorizzate le informazioni in una base di dati, o rispetto alle regole logiche con le quali è codificata la conoscenza su un dominio in una base di conoscenza. Il testo è, infatti, una forma di rappresentazione della conoscenza prodotta da e per le persone, perciò esso va processato in modo opportuno affinché il suo contenuto informativo possa essere acquisito, processato e reso disponibile automaticamente, e non soltanto tramite la lettura. Esistono due livelli di rappresentazione digitale del testo: il più superficiale è basato solo sulla *forma* del testo e non sul suo *significato*, diversamente da questo, il livello di rappresentazione dei testi più profondo e completo tiene conto del significato delle parole.

Prima di poter parlare di rappresentazione delle parole, bisogna effettuare una fase di pre-elaborazione del testo che ci permette di migliorare la qualità della rappresentazione.

2.1 PRE-ELABORAZIONE DEL TESTO

L’elaborazione in linguaggio naturale (NLP) è la capacità di un programma per computer di comprendere il linguaggio umano così com’è. Lo sviluppo delle applicazioni di NLP è difficile perché i computer richiedono tradizionalmente agli umani di “parlare” con loro in un linguaggio di programmazione che sia preciso, non ambiguo

e altamente strutturato, o attraverso un numero limitato di comandi vocali chiaramente enunciati. Il linguaggio umano, tuttavia, non è sempre preciso: è spesso ambiguo e la struttura linguistica può dipendere da molte variabili complesse, inclusi gergo, dialetti regionali e contesto sociale.

La difficoltà principale dell'elaborazione del linguaggio naturale è l'ambiguità che si può trovare in tutti i livelli del problema.

Un elenco di tipi di ambiguità che possono verificarsi sono:

ambiguità lessicale (un nome può essere anche un verbo)

ambiguità sintattica (interpretazione scorretta della frase in base al contesto)

ambiguità semantica (una parola può avere significati diversi)

ambiguità pragmatica (il significato della parola in un determinato contesto)

ambiguità di riferimento (un riferimento incerto).

Inoltre, ogni forma di ambiguità può interagire con le altre, rendendo il processo di interpretazione estremamente complicato. La presenza di ambiguità è proprio ciò che contraddistingue un linguaggio naturale da uno artificiale e rende la maggior parte delle tecniche utilizzate per l'analisi dei linguaggi di programmazione inefficaci.

Pertanto, bisogna eseguire diverse fasi di normalizzazione specifiche del dominio.

2.1.1 RIMOZIONE DELLE STOP WORDS

Le stop words sono parole molto comuni in una lingua o in un particolare ambito, al punto tale che il loro contenuto informativo è considerato molto basso; per questa ragione un testo si può rappresentare efficacemente senza comprenderle risparmiando spazio di rappresentazione e tempo di computazione. Per le lingue più comuni esistono liste di stop words che vengono usate per filtrare le parole rilevanti nei documenti.

2.1.2 TOKENIZATION

Il processo di Tokenization indica la suddivisione delle frasi in parole o "token", eliminando i segni di punteggiatura e i caratteri speciali. Anche in questo caso, l'euristica base con la quale si identificano i token è che essi sono generalmente separati da spazi; bisogna tuttavia tener conto di token in forme particolari, come le date e l'ora, i nomi propri di persona, ecc., che sono composti da più parti che non vanno considerate separatamente.

2.1.3 LEMMATIZATION & STEMMING

Nei documenti sono spesso presenti diverse forme di una parola che hanno dei significati simili (democrazia, democratico, democratizzazione). La lemmatizzazione è un'operazione di semplificazione dei token che trasforma ciascuno di essi nella sua forma grammaticale di base (ossia il suo lemma).

Lo stemming è un'operazione di semplificazione dei token più radicale: essa tronca il suffisso di ogni token riducendo il termine alla sua radice. Uno degli algoritmi più usati per compiere questa operazione è l'algoritmo di Porter.

2.1.4 POS TAGGING

Il Part Of Speech Tagging è l'operazione che associa a ciascun token la sua categoria grammaticale all'interno della frase. Questa operazione non è semplice, poiché molto spesso uno stesso termine può avere più di un significato, o perché alcuni elementi sintattici nella frase sono sottointesi nell'uso comune della lingua. La disambiguazione tra i diversi ruoli sintattici e l'individuazione di quello corretto avviene considerando sia le definizioni del termine considerato, che, soprattutto, le parole assieme alle quali occorre e il loro ruolo all'interno della frase.

L'operazione di POS Tagging è particolarmente utile per la costruzione di una rappresentazione della sintassi del testo e per l'individuazione della corretta semantica associata ai termini.

2.2 WORD REPRESENTATION

Nell'elaborazione del testo, le parole rappresentano caratteristiche discrete e categorizzabili. È necessario trovare un metodo per rappresentare i dati in modo tale che gli algoritmi possano utilizzarli per elaborare risultati. La mappatura da dati testuali a vettori con valori reali è chiamata estrazione delle caratteristiche.

Esistono diverse tecniche in grado di estrarre features da testo come ad esempio Bag Of Word, TF-IDF e Word Embedding.

2.2.1 BAG OF WORDS

Una delle tecniche più semplici per rappresentare numericamente il testo è **Bag of Words**. Attraverso tale tecnica creiamo l'elenco di parole uniche nel corpus di testo chiamato vocabolario. Quindi possiamo rappresentare ogni frase o documento come un vettore con ogni parola rappresentata con il conteggio della parola se presente e 0 se assente nel vocabolario.

Il modello Bag of Words è un metodo di rappresentazione di un testo che tiene conto del numero di occorrenze di ogni parola nel testo ma non delle posizioni che ciascuna occupa in esso. Questo però ha difetti, infatti nel caso in cui due documenti contengono le stesse parole ma in diverso ordine, quindi con significati diversi, saranno rappresentati nello stesso modo. Questo modello si basa sull'assunzione forte che la posizione di una parola nelle frasi del testo in cui compare non sia rilevante per stabilirne il significato e, dunque, per interpretare il contenuto informativo del testo. Sebbene questa assunzione sia errata, tale tecnica risulta utile nella ricerca di documenti rilevanti data una query.

2.2.2 TF-IDF

Un'altra rappresentazione simile e migliore che può misurare anche l'importanza di una parola in un documento prende il nome di **TF-IDF** ovvero il prodotto fra la

frequenza del termine e la frequenza del documento inversa nella collezione come indicato nella seguente formula.

$$w_{i,j} = tf_{i,j} \times \log \left(\frac{N}{df_i} \right)$$

$tf_{i,j}$ = number of occurrences of i in j
 df_i = number of documents containing i
 N = total number of documents

Il primo termine calcola la frequenza di termine (TF) che è matematicamente definita come la frequenza assoluta di occorrenza del termine nel documento ma generalmente, questa misura non viene direttamente impiegata per calcolare la rilevanza di un documento rispetto a una query, poiché porterebbe a far aumentare il peso di un termine in un documento in modo spropositato, facendo risultare il documento di una rilevanza rispetto alla query maggiore di quella reale (un documento in cui un termine appaia 10 volte non dovrebbe essere 10 volte più significativo di uno in cui esso appare una volta sola). Per questo si adotta diverse tecniche di normalizzazione, la più semplice ad esempio calcola il TF andando a dividere il numero di volte in cui una parola appare in un documento, per il numero totale di parole in quel documento; il secondo termine è la frequenza inversa dei documenti (IDF), calcolata come logaritmo del numero di documenti nel corpus diviso per il numero di documenti in cui appare il termine specifico.

Il TF-IDF viene spesso utilizzato nel recupero di informazioni e nell'estrazione di testo, poiché indica una misura statistica utilizzata per valutare l'importanza di una parola per un documento in una raccolta o in un corpus. L'importanza aumenta proporzionalmente al numero di volte in cui una parola appare nel documento ma è compensata dalla frequenza della parola nel corpus. Le variazioni dello schema di ponderazione TF-IDF sono spesso utilizzate dai motori di ricerca come strumento centrale per la valutazione e la classificazione della pertinenza di un documento in base a una query dell'utente.

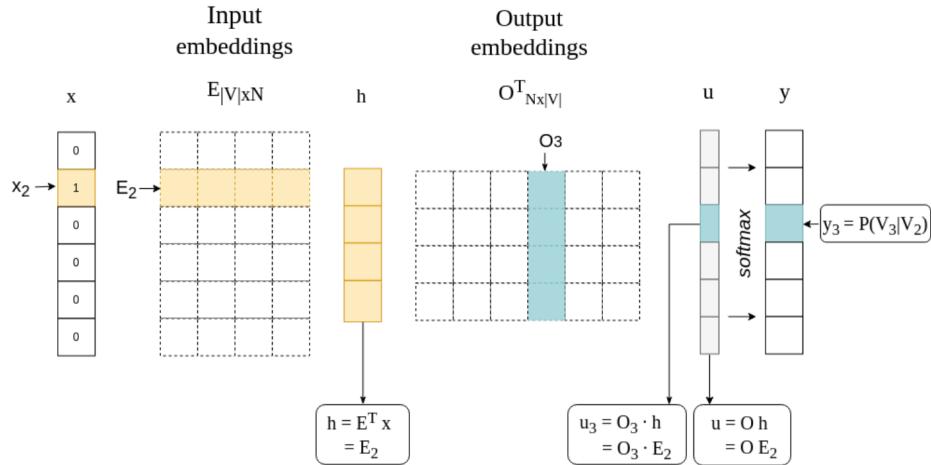
2.2.3 WORD EMBEDDING

Uno dei principali svantaggi delle tecniche viste è che eliminano l'ordine delle parole, ignorando quindi il contesto e a sua volta il significato delle parole nel documento. Il mantenimento del contesto delle parole nell'NLP è molto importante. Per risolvere questo problema usiamo un altro approccio chiamato **word embedding**, cioè una rappresentazione di testo in cui le parole che hanno lo stesso significato hanno una rappresentazione simile (in base ad esempio alla similarità del coseno). In altri termini, rappresenta le parole in un sistema di coordinate in cui le parole correlate, basate su un corpus di relazioni, sono messe più vicine rispetto a quelle meno correlate. Alcune delle tecniche più utilizzate per questo scopo sono: word2vec, GloVe e FastText.

Miklov e colleghi [1] nel 2013 hanno creato un software chiamato **word2vec**, esso rappresenta una rete neurale che permette di prendere in input un grande corpus di testo e produrre come output uno spazio vettoriale con ogni parola univoca assegnata a un vettore corrispondente nello spazio. Sono stati ideati due diversi approcci: CBoW (Continuous Bag of Word) e CSG (Continuous Skip-Gram) CBoW mira a prevedere la parola in base al contesto di input, cioè in base al testo a sinistra e a destra della parola stessa. Esso risulta semplice e veloce da implementare.

CSG è l'opposto del precedente, in quanto qui, data una parola, si cerca di capire quale è il contesto nel quale viene usato più spesso. A differenza del modello precedente, quest'ultimo è più difficile da implementare ma si comporta meglio con parole meno frequenti. In termini di architettura [19], Skip-gram è una semplice rete neurale con un solo livello nascosto. L'input alla rete è una rappresentazione vettoriale codificata a caldo di una parola di destinazione: tutte le sue dimensioni

sono impostate su zero, a parte la dimensione corrispondente alla parola di destinazione.



L' algoritmo Global Vectors for Word Representation, o **GloVe** è un'estensione del metodo word2vec per l'apprendimento efficiente dei vettori di parole. GloVe nasce grazie Pennington et al., che in [2] sostengono che l'approccio di scansione online utilizzato da word2vec è subottimale in quanto non sfrutta appieno le informazioni statistiche globali relative alle co-occorrenze di parole. Infatti, proprio come si può leggere all'interno del loro articolo, *"Il modello produce uno spazio vettoriale con una sottostruttura significativa, come dimostra la sua prestazione del 75% su un compito di analogia di parole recenti. Supera anche modelli correlati su attività di similarità e riconoscimento di entità con nome."*. In poche parole, GloVe costruisce una matrice esplicita di parole-contesto o di co-occorrenza usando le statistiche sull'intero corpus di testo. Queste matrici rappresentano solitamente l'occorrenza o l'assenza di parole in un documento. Il risultato è un modello di apprendimento che può generare in genere migliori word embedding rispetto a word2vec.

L'algoritmo **FastText** è un altro metodo di word embedding che rappresenta un'estensione del modello word2vec. In questa tecnica, invece di imparare direttamente i vettori per le parole, viene rappresentata ogni parola come un n-gram di caratteri. Ad esempio se si sceglie $n = 3$, la parola “artificiale” sarà rappresentata in questo modo: <ar, art, rti , tif, ifi, fic, ici, cia, ial, ale, le>. Come i suoi ideatori dicono nell'articolo in cui fastText è stato presentato [8], grazie a tale rappresentazione è possibile catturare il significato di piccole parole e capire quali sono i loro suffissi e prefissi. Una volta rappresentata la parola in BoC (Bag of Character) di n-gram caratteri, viene addestrato un modello skip-gram in grado di capire quali sono gli embedding. Quindi fastText può essere considerata come word2vec solo che anziché considerare direttamente le parole, si considerano le sottoparole. Inoltre, Mikolov at al [8] indicano che la loro tecnica funziona molto bene nei casi di parole rare. Infatti, a differenza di word2vec e GloVe, essa può rappresentare una parola anche se non presente nel vocabolario poiché suddividendola in n-gram è possibile risalire al suo embedding. Questo risulta essere un grande vantaggio rispetto alle tecniche precedenti.

Le tecniche viste fino ad ora, funzionano molto bene con la lingua inglese. La lingua italiana risulta morfologicamente complessa e profondamente diversa dalla lingua inglese. Diverse soluzioni sono state implementate a tale scopo.

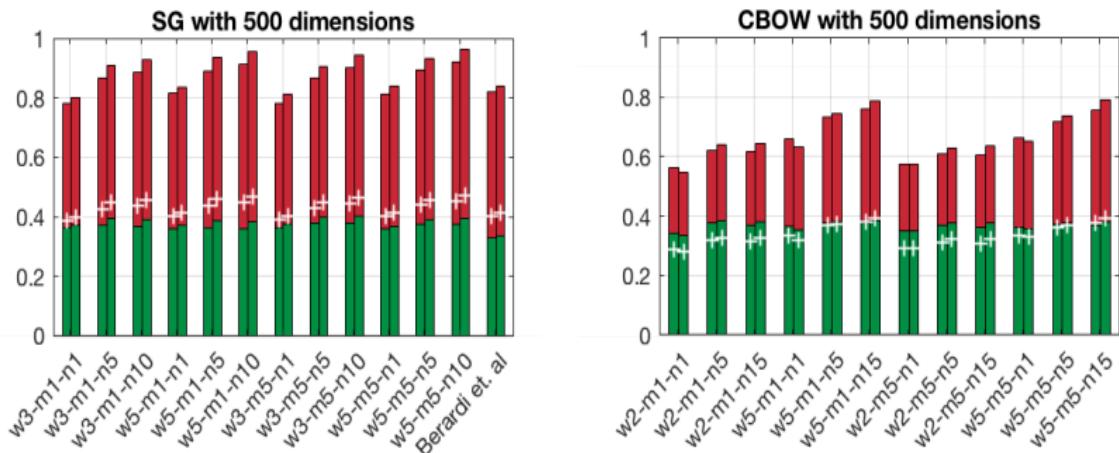
Rocco Tripoli e Stefano Li Pira [9] hanno testato due modelli di rete neurale per effettuare la rappresentazione delle parole, SG e CBOW, addestrando solo su una parte di Wikipedia italiano. Hanno confrontato i risultati dei due modelli utilizzando 12 combinazioni di iperparametri comprendenti la dimensione del vettore (dim), la dimensione della finestra del contesto di una parola (w), il numero di occorrenze

della parola meno frequente (m) e il numero di esempi negativi (n). Le varie combinazioni sono le seguenti:

HP	SG	CBOW
<i>dim</i>	200, 300, 400, 500	200, 300, 400, 500
<i>w</i>	3, 5	2, 5
<i>m</i>	1, 5	1, 5
<i>n</i>	1, 5, 10	1, 5, 15

Dopo aver effettuato alcune operazioni come lowercase e filtri in accordo con gli iperparametri, il corpus risultava avere 994.949 frasi, risultanti in 470.400.914 token.

Partendo da un dataset contenente 19.791 domande con 19 diverse relazioni sia sull'aspetto morfologico (barra verde) che sull'aspetto semantico (barra rossa) della parola, Rocco Tripoli e Stefano Li Pira adottando un semplice test di similarità (formule di analogia 3COSADD e 3COSMUL in grado di catturare le diverse relazioni fra le varie parole) degli embedding prodotti con embedding creati da Berardi et al (2015) sullo stesso dataset, hanno dimostrato che l'aumento del numero di dimensioni e il numero di esempi negativi migliorano le prestazioni di entrambi i modelli, in particolare utilizzando Skip Gram con dimensione 500, quindi SG500.



Andrea Cimino et al. hanno presentato alla conferenza EVALITA nel 2018, ovvero Evaluation of NLP and speech tools for Italian, e subito dopo pubblicato nell'articolo [12], tre set di word embedding generati a partire da due corpus differenti, testati sui task di ABSITA, GxG, HaSpeeDe ed IronITA.

Il primo prende il nome di itWaC, addestrato su un corpus di miliardi di parole costruito dal Web che limita la scansione al dominio .it e utilizza come media le parole a media frequenza del corpus di Repubblica e gli elenchi del vocabolario italiano di base. Il secondo si basa su Twitter è invece addestrato su 46.935.207 tweet. Entrambi questi sistemi costituiscono dei word embedding di dimensione 128, una finestra di 5 parole, e cbow.

Ogni voce dei lessici mappa una coppia (parola, POS) per incorporare la parola associata, permettendo di attenuare i problemi di polisemia che possono portare a risultati peggiori nella classificazione. Inoltre, distingue tra tutte le parole maiuscole e le parole non maiuscole (es.: "mai" vs "MAI"), poiché tutte le parole maiuscole vengono solitamente utilizzate in contesti negativi.

È importante notare che i word embedding codificano ogni parola in un vettore che cattura una sorta di relazione e somiglianza tra le parole all'interno del corpo del testo. Ciò significa che anche le variazioni di parole come caso, ortografia, punteggiatura e così via verranno automaticamente apprese. A sua volta, ciò può significare che alcune delle operazioni di pulizia del testo descritte sopra potrebbero non essere più necessarie o potrebbero essere automaticamente effettuate.

2.3 INTENT DETECTION

Una volta rappresentate le query attraverso una delle tecniche viste precedentemente, sarà necessario determinare l'Intent insito all'interno.

In letteratura per la rilevazione dell'intent ci si basa principalmente su diversi metodi di apprendimento supervisionato, dove, date delle features di input, features target (obiettivo) e un insieme di esempi di training per cui siano disponibili i loro valori, bisogna predire i valori dei target per nuovi esempi, noti solo i valori di quelle di input; un approccio differente lo abbiamo con l'apprendimento non supervisionato dove non sono definiti i valori per le feature target degli esempi di input e quindi il sistema deve calcolarli automaticamente.

In queste tecniche è importante definire se l'apprendimento deve essere effettuato offline (gli esempi sono disponibili a priori) oppure online (l'acquisizione degli esempi avviene nel tempo, mentre avviene il processo di learning. Chiamata tecnica Active Learning).

Quasi tutti gli approcci standard all'apprendimento supervisionato sono stati applicati nella classificazione degli intent, come è possibile verificare in [3], in cui nello specifico vengono valutate la maggior parte delle tecniche di rappresentazione del testo e la maggior parte degli algoritmi di intent detection, successivamente testati sullo stesso dataset, Snips 2017-06-custom intent-engines. In questo dataset ci sono 7000 esempi di training e 700 esempi di testing. Le categorie possibili di intent sono 7. I risultati vengono valutati attraverso una metrica di misura F1-Score e riportati nella seguente tabella:

Classifiers	Count	Tfidf	Count-Lsa	Tfidf-Lsa	SG25-Avg	SG25-Idf Avg	SG-512 Avg	SG-512 Idf Avg
Linear SVM	91.18	91.43	91.16	91.17	93.58	94.15	92.50	94.82
Logistic Regression	91.60	91.30	91.20	92.49	92.72	94.73	91.42	94.17
K Neighbors	87.42	89.91	89.72	90.28	93.16	92.80	90.88	91.76
Random Forest	91.28	91.56	87.71	90.15	93.56	93.64	91.44	93.80
SVM	84.61	86.49	88.71	84.14	91.66	91.91	92.52	92.93
Neural Networks	93.58	93.67	93.74	93.58	94.53	95.01	94.93	95.16
Cosine Similarity	92.83	85.01	90.58	91.18	95.42	93.90	94.18	93.44
Decision Tree	90.88	90.89	89.21	89.68	91.96	91.88	91.46	92.70

Di seguito verranno presentate a livello teorico la maggior parte delle tecniche presenti a stato dell'arte, soffermandosi più nello specifico sullo studio delle reti neurali.

2.3.1 CLUSTERING

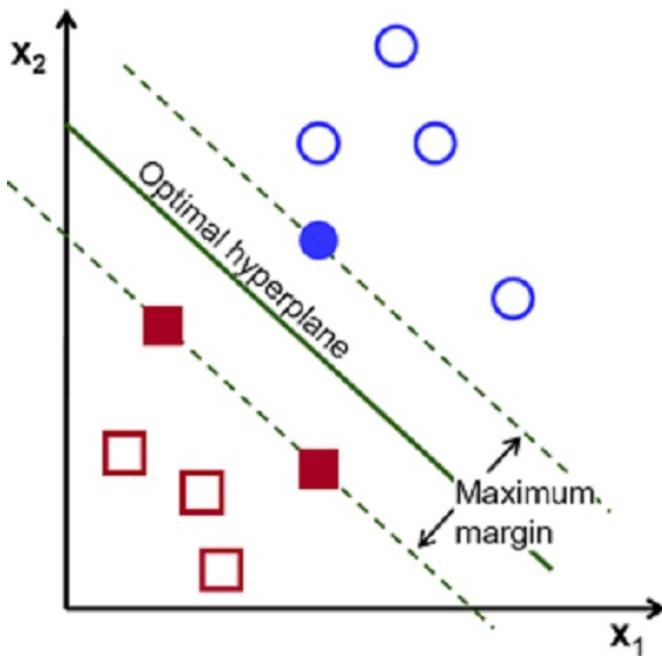
Un approccio che fa riferimento invece all'apprendimento non supervisionato è il **clustering**: esso ha lo scopo di partizionare un set di dati in cluster (classi) in modo tale da individuare gruppi significativi che esistono nel set di dati. Fra gli algoritmi più utilizzati rientrano il K-means, Self-Organizing Map (SOM) o EM.

Colin Shengcai Zhao ha presentato un sistema di riconoscimento di intent da una query utilizzando i cluster, in [26], in cui sono descritti un metodo e un sistema per utilizzare il social tagging per identificare l'intento dell'utente di un motore di ricerca. Un motore di ricerca seleziona un insieme di pagine rilevanti per una query. Il motore determina i vicini di queste pagine e raggruppa i vicini in cluster di attualità. Per ciascun cluster, il motore determina i tag che la comunità di utenti ha frequentemente associato alle pagine di quel cluster. Per ogni cluster, il motore abbina i tag dominanti di quel cluster con le frasi in vari elenchi di intenti. Abbinando i tag di un determinato cluster a vari elenchi, viene determinato un intento corrispondente a quel particolare cluster. Per ogni intento del cluster, il motore di ricerca può presentare, con i risultati della query, tipi di contenuto che corrispondono in particolare a tale intento (ad esempio, una mappa per un intento di posizione, possibilmente insieme alle indicazioni stradali). Inoltre, o in alternativa, per ciascuno di tali intenti, il motore di ricerca può formattare i risultati della query in un modo adatto all'intento.

2.3.2 SVM

Un altro approccio sarebbe l'utilizzo di combinazioni di macchine vettoriali e di supporto lineare (**SVM**), come testato in [4], più nello specifico viene addestrata una macchina SVM per ogni categoria e gli esempi di addestramento positivo

(negativo) sono gli esempi che appartengono alla categoria di destinazione (rispettivamente che non appartengono).



L'addestramento di un classificatore SVM implica l'individuazione di un iperpiano, come superficie di decisione, che separa gli esempi di addestramento positivi da quelli negativi con il margine maggiore. I vettori che formano l'iperpiano sono conosciuti come vettori di supporto. Tale iperpiano è quindi usato per prevedere le classi delle nuove istanze.

2.3.3 CLASSIFICATORE BAYESIANO

Un altro approccio è l'uso di un classificatore Bayesiano, come ad esempio **Naive Bayes** oppure **Logistic Regression**. Il classificatore Naive Bayes è un classificatore lineare noto per essere semplice e molto efficienti. Il modello probabilistico dei classificatori Naive Bayes si basa sul teorema di Bayes, e il termine Naive è un aggettivo che deriva dal presupposto che le caratteristiche di un set di dati sono reciprocamente indipendenti. In pratica, il presupposto di indipendenza è spesso violato, ma i classificatori Naive Bayes tendono a funzionare molto bene sotto questo presupposto irrealistico. Soprattutto per campioni di piccole dimensioni,

può superare le alternative più potenti. Un'altra applicazione del classificatore bayesiano avviene attraverso il classificatore di regressione logistica è un modello supervisionato che viene utilizzata quando la variabile di output è categorica, ovvero non indica ambiguità. L'idea della regressione logistica consiste nel trovare la probabilità condizionata dell'output dato il suo input.

La regressione logistica utilizza la funzione Logit o la funzione logodds per il calcolo della probabilità condizionale. Di solito viene utilizzato per testi di grandi dimensioni. In [13] viene descritto un approccio bayesiano alla regressione logistica che evita il sovradattamento; essa ha un'efficacia di classificazione simile a quella dei migliori metodi pubblicati ed è efficiente sia durante la fase di training che al momento della previsione. La chiave del loro approccio sta nell'uso di una distribuzione di probabilità precedente che favorisce la scarsità nel modello adattato, insieme ad un algoritmo di ottimizzazione e implementazione basato sul modello descritto precedentemente.

2.3.4 CASE-BASED

Un altro approccio all'Intent Detection lo abbiamo attraverso l'**apprendimento basato sui casi**: gli esempi di addestramento, cioè i casi, sono memorizzati e accessibili per risolvere un nuovo problema. Per ottenere una previsione per un nuovo esempio, i casi simili o prossimi ad esso vengono utilizzati per prevedere il valore delle sue features target (basato su k-NN).

Questo è ad un estremo del problema dell'apprendimento in cui, a differenza degli alberi decisionali e delle reti neurali, è necessario eseguire relativamente poco lavoro offline e praticamente tutto il lavoro viene eseguito al momento dell'interrogazione. Sarà necessaria una misura di distanza (euclidea) calcolata tra i vari esempi che indicherà la misura di similarità.

In [14] vengono proposte molte delle tecniche e vengono valuti i pro e contro di ognuna; la tabella riportata indica solo alcune delle tecniche analizzate.

Technique	Concept	Merits	Demerits	Applications
k-d tree nearest neighbor (kdNN) [21]	To divide the training data sets into two halves	1.Perfect balanced trees are formed 2.It is fast and simple	1.Computational complexity 2.Exhaustive search is required 3.Chance of misleading the points as it blindly splits the points into two halves.	Multidimensional data points.
Clustered k nearest neighbor [40]	To select the nearest neighbor from the clusters	1.Overcome defects of uneven distributions of training samples 2.Robust in nature	1.Selection of threshold parameter is difficult before running algorithm 2.Biased by value of k for clustering	Text Classification
k Nearest Neighbor (kNN) [9]	To find the nearest neighbor based on 'k' value	1.Training is very fast 2.Simple and easy to learn 3.Robust to noisy training data 4.Effective if training data is large 5.It is symmetric.	1. Biased by value of k 2.Computation Complexity 3.Memory limitation 4.Being a supervised learning lazy algorithm i.e. runs slowly 5.Easily fooled by anomalies	Large sample data
Weighted k nearest neighbor (WkNN) [33]	To assign weights to neighbors based on distance calculated	1. Overcomes limitations of kNN by assigning equal weight to k neighbors implicitly. 2. Uses all training samples not just k. 3. Makes the algorithm global one	1.Computational complexity increases in calculating weights 2.Slow in execution	Large sample data
Aggregate kNN [49]	To use aggregate function for finding the nearest neighbor	1.Provides memory-resident queries and cost models that accurately predict their performance in terms of node accesses 2.Approximate result can be obtained for	1. Cost for evaluating the disk-resident query model is high. 2. Lazy algorithm	Spatial data set

2.3.5 ALBERI DECISIONALI

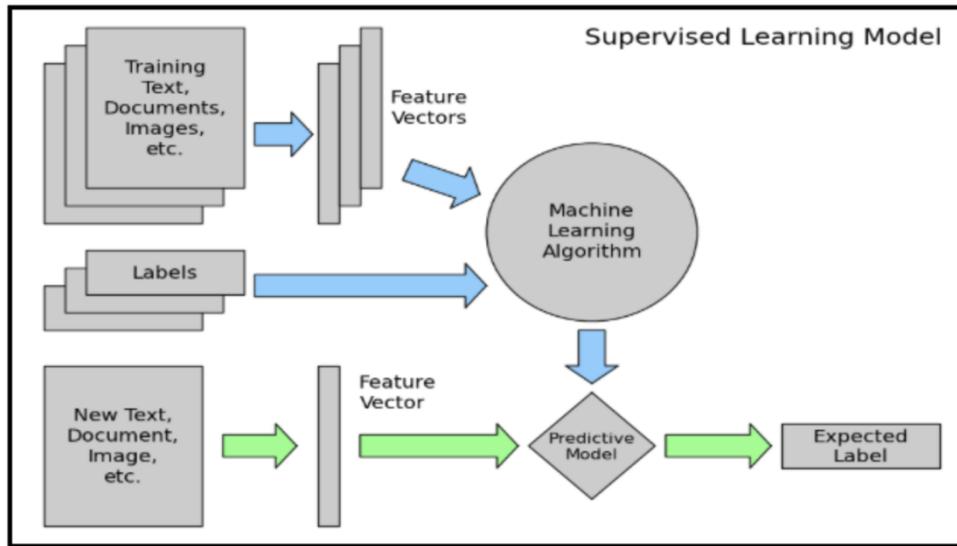
Un algoritmo che applica tale teoria è il ANN (Approximated Nearest Neighbors).

Un altro approccio si basa su **alberi decisionali** [5] : essi costituiscono una sequenza di condizioni in una struttura ad albero. La radice e i nodi interni contengono condizioni di test degli attributi e i nodi terminali sono assegnati a un'etichetta di classe. Gli algoritmi per la costruzione di alberi decisionali di solito funzionano dall'alto verso il basso, scegliendo una variabile ad ogni passaggio che meglio divide l'insieme di elementi. Gli algoritmi per generare alberi decisionali funzionano selezionando una variabile ad ogni passaggio. Approcci come il information gain,

Gini-Index vengono utilizzati per trovare la variabile più adatta per lo stesso. Questa tecnica risulta comoda quando si trattano features booleane quindi semplicemente l'albero si divide in base a cosa rendono true e a cosa rendono false gli esempi di addestramento. In caso di features non booleane la situazione si complica leggermente, potremmo considerare una tecnica di multiway split nonché una suddivisione in più vie dell'albero ma questo renderebbe troppo specifici i rami e potrebbe a volte non capire come classificare un esempio, oppure una tecnica di suddivisione del dominio in due sottoinsiemi così da ritrovarsi sempre o in un caso o nell'altro. Un problema di questa tecnica è il sovraccarico. Per evitare ciò, è possibile l'utilizzo di un classificatore random forest, nonché un ensemble learning, ovvero un sistema di addestramento che utilizza più classificatori per decidere, in cui tutti i classificatori sono degli alberi decisionali.

2.3.6 RETI NEURALI

Un altro approccio, molto utilizzato negli ultimi tempi, si basa su **reti neurali o deep learning**: sono una rappresentazione di destinazione popolare per l'apprendimento. Queste reti sono ispirate ai neuroni del cervello. Le reti neurali artificiali in genere contengono molti meno neuroni rispetto a quelli che si trovano nel cervello umano, e tali neuroni artificiali, chiamati unità, sono molto più semplici delle loro controparti biologiche.



Le reti neurali hanno in genere diverse features di input ma possono avere anche più features-obiettivo. Gli input si alimentano in strati di unità nascoste, che possono essere considerate come caratteristiche che non vengono mai osservate direttamente, ma che sono utili per la predizione. Ogni strato indipendentemente dagli altri, valuta l'input e lo classifica attraverso gli esempi a disposizione. la somma di tutti questi output moderata anche dai pesi per ogni risultato dei diversi "neuroni" sarà il risultato di output finale. La parte fondamentale delle reti neurali è lo strato nascosto (hidden layer) che elabora l'output di un altro neurone per produrre una risposta più specifica aggiungendo features fino ad arrivare all'output finale senza essere visto dall'esterno. La formula da uno strato a quello successivo è questa equazione:

$$o_j = f \left(\sum_i w_{i,j} a_i + b_i \right)$$

Il livello con i nodi a funge da input per il livello con i nodi o . Per calcolare i valori per ciascun nodo di output, dobbiamo moltiplicare ciascun nodo di input per un peso w e aggiungere un bias b .

Tutti questi devono quindi essere sommati e passati a una funzione f . Questa funzione è considerata la funzione di attivazione e ci sono varie funzioni che possono essere utilizzate a seconda del livello o del problema. È generalmente comune utilizzare un'unità lineare rettificata (ReLU) per i livelli nascosti, una funzione sigmoide per il livello di output in un problema di classificazione binaria o una funzione softmax per lo strato di output di problemi di classificazione multi-classe.

Ogni layer è implementato come un modulo separato che implementa in avanti la predizione e indietro la BP (Back Propagation) cioè passa l'importanza dei pesi, aggiornata man mano, anche ai layer precedenti: L'algoritmo prima inizializza i pesi con valori casuali che vengono modificati durante l'addestramento attraverso BP. Questo viene fatto usando metodi di ottimizzazione (chiamati anche ottimizzatori) come la discesa del gradiente, utilizzata nello specifico per ridurre l'errore tra l'output calcolato e quello desiderato (chiamato anche output di destinazione). L'errore è determinato da una funzione di perdita di cui vogliamo ridurre l'entità con l'ottimizzatore come ad esempio la funzione di perdita dell'entropia.

Nell'addestrare ogni neurone esistono due varianti dell'algoritmo di backpropagation:

- Per pattern: aggiorna i pesi in modo backward partendo dal livello di output, utilizzando la delta rule generalizzata:
- Per epoche: chiamata modalità batch, i pesi sono aggiornati solo dopo aver processato tutti gli esempi. Il processo di apprendimento continua di epoca in epoca fino alla condizione di arresto. Sono stati proposti due criteri di arresto:
 - Cambiamento errore quadratico medio totale: si considera una situazione di convergenza quando il tasso di cambiamento dell'errore

quadratico medio per epoca è sufficientemente piccolo (o non cambia proprio), solitamente nel range di cambiamento di [0.1, 0.01].

- Basato sulla generalizzazione: si partiziona il training set in una piccola porzione chiamata validation set, utilizzata per testare la generalizzazione della rete dopo ogni epoca. Se le performance sono adeguate ci si ferma.

Ricapitolando, in una rete neurale ci sono diversi parametri che devono essere decisi:

- Valore iniziale dei pesi: solitamente si sceglie in modo random tra $[-1, 1]$ oppure $[-\frac{1}{2}, \frac{1}{2}]$
- Learning rate η : la scelta di η dipende dall'applicazione, solitamente si sceglie tra 0.1 e 0.9, oppure il valore viene adattato ad ogni iterazione (come detto prima)
- Numero di livelli nascosti e numero di neuroni
- Numero di esempi da usare per il training set

I dataset iniziale è stato suddiviso in due principali sottoinsiemi:

- training set: usato unicamente per addestrare la rete (cambiare i pesi), solitamente è il 80 % del set iniziale;
 - validation set: sottoinsieme del training set, generalmente il 15/20% del training set. Viene usato sia per testare la generalizzazione sia per settare gli iperparametri;
 - estimation set: restante dell'insieme usato per il training del modello.

- o test set: tutto ciò che non fa parte del training set (generalmente possiamo dimenticarcene, lo useremo solo per valutare il funzionamento su esempi mai visti).

Un comune problema a cui si può andare incontro durante l'apprendimento della funzione è l'overfitting. Questa condizione particolare si verifica nel momento in cui il sistema si specializza sul training set degli esempi da cui apprende talmente tanto da non riuscire ad apprendere qualsiasi altro input esterno al training set.

Per poter ovviare a questo problema sono comuni due soluzioni:

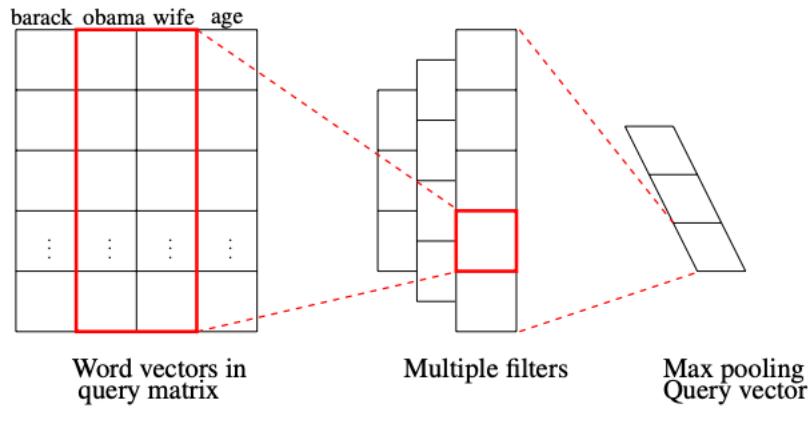
- ridurre il numero di features apprese
- effettuare una regolarizzazione dei parametri

Per evitare ciò, è possibile ricorrere ad alcuni miglioramenti. Prima di tutto bisogna notare che i layer completamente collegati sono inclini a un overfitting, soluzione a ciò potrebbe essere il livello di Dropout, proposto da Hinton et al. come un regolarizzatore che imposta in modo casuale la metà delle attivazioni sugli strati completamente collegati a zero durante l'allenamento. Ha migliorato la capacità di generalizzazione e in gran parte impedisce il sovradimensionamento. Esiste anche un'altra strategia chiamata Global Max Pool per sostituire i tradizionali livelli completamente collegati nella reti. L'idea è di generare una mappa delle caratteristiche per ciascuna categoria corrispondente dell'attività di classificazione nell'ultimo livello. Invece di aggiungere livelli completamente connessi in cima alle mappe delle caratteristiche, prendiamo la media di ciascuna mappa delle caratteristiche e il vettore risultante viene immesso direttamente nel livello softmax.

Il fascino delle reti neurali sta nel fatto che i layer nascosti non devono essere definiti dal progettista infatti ciò che i livelli nascosti rappresentano è un risultato di apprendimento.

Per l'NLP vengono principalmente utilizzate reti neurali RNN e CNN.

Le **reti CNN** ovvero “**reti neurali convoluzionali**” (CNN) sono delle reti neurali che trovano vasta applicazione nel campo della Computer Vision, ma che hanno trovato riscontri positivi in determinate applicazioni dell'Elaborazione del Linguaggio Naturale. Esse sono chiamate così poiché si basano sulla convoluzione. Come descritto in [21], questo è un meccanismo che permette di far scorrere un filtro (o kernel) sull'input e sono costituite da tre specifiche tipologie di strati:



Esempio di applicazione di una CNN ad una query presentata in [22]

strato di convoluzione: caratteristico di questo tipo di rete neurale, è ispirato dai processi biologici per l'analisi visiva negli organismi viventi. Lo strato di neuroni che si occupa della convoluzione divide i dati in vari frammenti sovrapposti, che sono in seguito analizzati per individuare le particolarità che lo caratterizzano, trasferendo l'informazione allo strato seguente sottoforma di una “feature map” contenente le relazioni tra neuroni e particolarità. Per fare ciò, ad ogni posizionamento sull'input, vengono moltiplicati i pesi del filtro con il contenuto dell'input e ne viene calcolata

una somma. Il risultato di questa operazione di convoluzione è un nuovo vettore/matrice (feature map), di dimensioni inferiori rispetto all'input, che contiene una rappresentazione dello stesso ma di più alto livello. È possibile settare la larghezza di convoluzione per indicare di quanto deve essere stretta rispetto all'originale.

strato di pooling: è un processo alquanto comune nelle reti neurali, che consiste nel ridurre la dimensione dei dati generalizzando, in modo da rendere più rapida l'analisi senza perdere troppa precisione. Nel caso di un'immagine, il processo è molto simile a una riduzione di qualità dei pixel. Tale strato quindi permette di semplificare le informazioni estratte da uno strato di convoluzione. Il grado di semplificazione dipende dal valore dell'offset per lo spostamento della finestra ("stride"). Un algoritmo di pooling usato frequentemente è il max pooling, che preleva il valore più alto sulla finestra.

strato di normalizzazione: permette di evitare le anomalie dovute agli strati precedenti. La funzione più utilizzata, poiché la più rapida, è chiamata ReLu ovvero Rectified Linear Unit

$$f(x) = x^+ = \max(0, x),$$

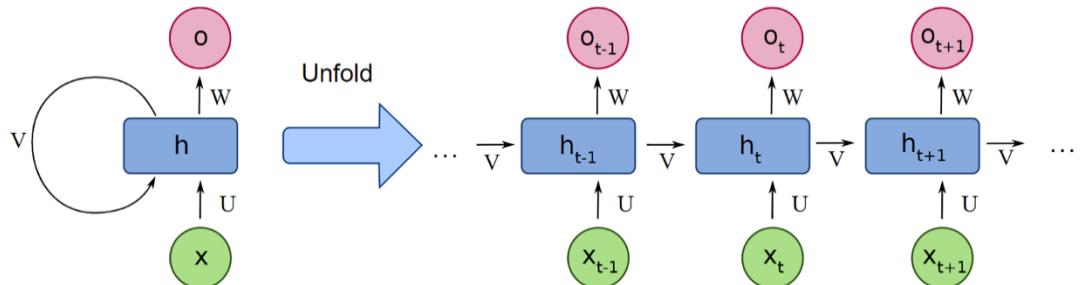
È presente anche un quarto strato chiamato full connection layer ovvero lo strato di connessione totale. È l'ultimo strato nascosto della rete neurale, nel quale tutti gli input dei vari neuroni sono messi insieme, permettendo alle particolarità di essere identificate. Esso è necessario alla fine del processo per classificare l'input rappresentato con un livello di astrazione più alto. In questo strato tutti i nodi di strati consecutivi sono connessi tra loro.

È importante notare che una rete neurale convoluzionale per il testo opera esclusivamente su due dimensioni, dato che i filtri devono muoversi solo rispetto

all'asse temporale. Le reti convoluzionali presentano alcuni vantaggi in termini di performance: sono più parallelizzabili delle reti ricorrenti, visto che lo stato della rete, ad ogni passo, dipende unicamente dal contesto locale (attraverso l'operatore di convolution) anziché dall'insieme degli stati passati, come per le RNN. Questo però risulta essere anche un difetto in alcune query poiché in questo modo perderebbe informazioni importanti di contesto.

Per ovviare al problema del contesto è possibile ampliare il campo di ricezione delle CNN utilizzando un operatore di convolution dilatato.

Le **reti RNN** ovvero “**rete neurale ricorrente**” sono una particolare tipologia di rete che ha memoria di quello che ha processato nel breve termine: esse infatti prendono in input sia l'esempio attuale che quello che hanno percepito precedentemente in maniera ciclica, in modo tale da poter considerare più di un esempio e avere più contesto e precisione nella decisione.



Facendo un esempio di rete RNN per predire una parola alla fine di una frase, dato un corpus grande dal quale imparare e rappresentata ogni parola con dei word embedding, utilizzeremo principalmente 3 formule:

$$h_t = f(W^{hh}h_{t-1} + W^{hx}x_t)$$

contiene le informazioni sulle parole precedente. Infatti, h_t è calcolato usando il precedente vettore h_{t-1} e il vettore di parola corrente x_t . Applichiamo anche una funzione di attivazione non lineare f (di solito tanh o sigmoid) alla sommatoria finale. È accettabile assumere che h_0 sia un vettore di zeri.

$$y_t = \text{softmax}(W^{(S)}h_t)$$

calcola il vettore di parole previsto in un determinato intervallo di tempo t utilizzando ad esempio la funzione softmax per produrre un vettore $(V, 1)$ con tutti gli elementi che sommano fino a 1. Questa distribuzione di

$$J^{(t)}(\theta) = \sum_{i=1}^{|V|} (y_{t_i}' \log y_{t_i})$$

probabilità ci fornisce l'indice della parola successiva più probabile dal vocabolario. Utilizza la funzione di perdita di entropia incrociata in ogni fase del tempo t per calcolare l'errore tra la parola prevista e quella effettiva.

I pesi W sono matrici inizializzate con elementi casuali, regolati utilizzando l'errore dalla funzione di perdita. Facciamo questa regolazione usando l'algoritmo di back-propagation.

Una volta ottenuti i pesi giusti per i dati di training, sarà possibile effettuare una nuova previsione.

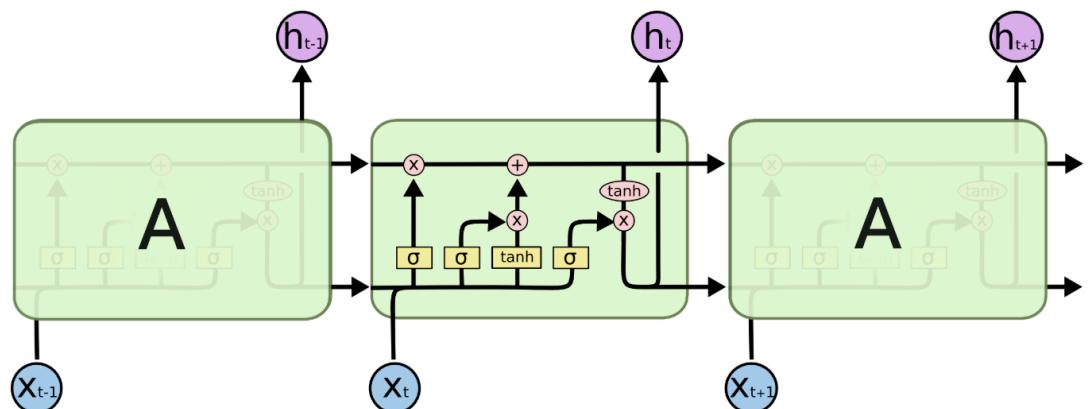
Queste informazioni vengono quindi composte tra loro per poter ottenere un qualche genere di correlazione tra eventi prossimi tra loro. Queste correlazioni vengono chiamate "dipendenze a lungo termine" perché ogni evento è funzione degli eventi che sono preceduti.

Un esempio molto pratico di come una rete neurale ricorrente possa funzionare, possiamo averlo immaginando di scrivere un messaggio di testo e di scrivere la parola ciao senza però scrivere la o finale, quindi solo cia. In tal caso la rete dovrebbe cercare di capire quale delle parole del vocabolario sia la più adatta al

conto, per questo tiene conto di tutte le lettere e dell'ordine in cui compaiono in modo tale da predire la lettera giusta.

Le RNN "pure" presentano un grosso problema ovvero non riescono a ricordare parole viste molto tempo prima ma tendono sempre a ricordare le più recenti, infatti sono state rapidamente rimpiazzate dalle **LSTM** (Long Short Term Memory) che differiscono per il fatto che gli aggiornamenti del livello nascosto vengono sostituiti da celle di memoria appositamente create che permette di ricordare le dipendenze a lungo termine.

Come spiegato in [23], la chiave del funzionamento per LSTM è lo stato della cella, ovvero la linea orizzontale che attraversa la parte superiore del diagramma.



Lo stato della cella è un po' come un nastro trasportatore. Corre lungo tutta la catena, con solo alcune interazioni lineari minori. È molto facile per le informazioni fluire lungo il percorso invariato.

L'LSTM ha la capacità di rimuovere o aggiungere informazioni allo stato della cella, regolato con cura dalle strutture chiamate gate.

I gate sono un modo per consentire l'informazione. Sono composti da uno strato di rete neurale sigmoide e da un'operazione di moltiplicazione puntuale.

Lo strato sigmoideo emette numeri compresi tra zero e uno, descrivendo la quantità di ogni componente che deve essere lasciata passare. Un valore pari a zero significa "non lasciar passare nulla", mentre il valore di uno significa "lascia che tutto passi!"

Un LSTM ha tre di questi gate, per proteggere e controllare lo stato della cella.

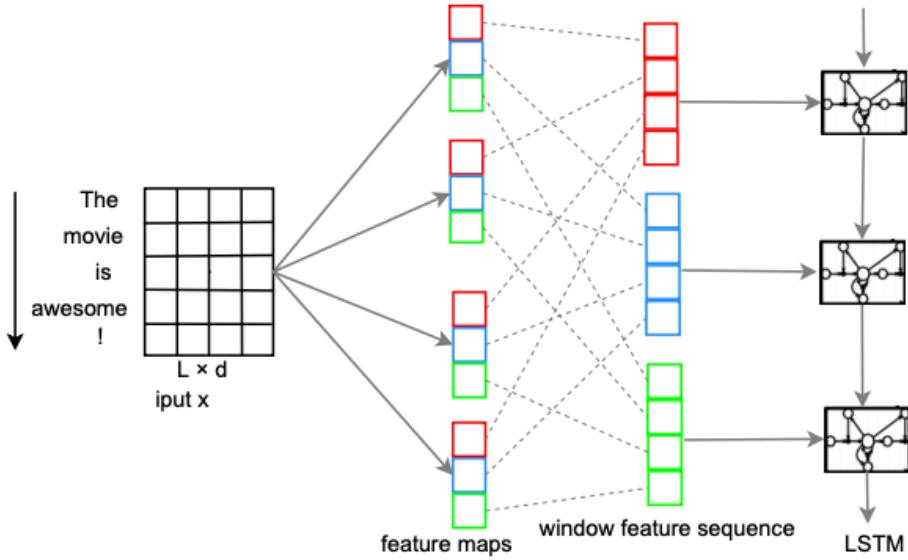
LSTM nel suo nucleo, conserva le informazioni dagli input che lo hanno già attraversato utilizzando lo stato nascosto, quindi conserva solo le informazioni del passato.

È possibile implementare una architettura di LSTM bidirezionale. Essa eseguirà gli input in due modi, uno dal passato al futuro, come nel caso delle LSTM unidirezionali, e uno dal futuro al passato. Quindi nell'LSTM che si esegue all'indietro si conservano le informazioni dal futuro.

Utilizzando i due stati nascosti combinati è possibile in qualsiasi momento conservare le informazioni dal passato e dal futuro.

Questa tecnica è molto utile nel caso di domande molto complicate, ma in ogni caso, i BiLSTM mostrano risultati molto buoni in quanto possono comprendere meglio il contesto.

Come evidenziato da Chunling Zhou et al. in [15], è possibile combinare i punti di forza di entrambe le architetture l'operazione di convolution può essere usata per velocizzare le LSTM. In [15] viene proposto un nuovo e unificato modello chiamato C-LSTM per la rappresentazione della frase e la classificazione del testo.



Per la classificazione del testo, considerano l'output dello stato nascosto nell'ultima fase temporale di LSTM e aggiungono uno strato di softmax in cima, fornendo l'intero modello minimizzando l'errore di entropia.

$$L(\mathbf{x}^{(i)}, y^{(i)}) = \sum_{j=1}^k 1\{y^{(i)} = j\} \log(\hat{y}_j^{(i)}),$$

Usando un dataset di 5452 domande classificate, e 500 non classificate, Chunting Zhou et al. hanno addestrato e testato tale sistema considerando 6 categorie possibili. Prendendo alcuni risultati anche da TREC Sono riusciti ad ottenere un'accuratezza del 94.6 % come mostrato dalla tabella in cui vengono presentate anche altre tecniche valutate sullo stesso dataset.

Model	Acc
SVM	95.0
Paragraph Vector	91.8
Ada-CNN	92.4
CNN-non-static	93.6
CNN-multichannel	92.2
DCNN	93.0
LSTM	93.2
Bi-LSTM	93.0
C-LSTM	94.6

CAPITOLO 3 - STATO DELLA PRATICA

La digitalizzazione ha cambiato il modo in cui elaboriamo e analizziamo le informazioni. Esiste un aumento esponenziale della disponibilità di informazioni online. L'idea è di creare, analizzare e riportare rapidamente le informazioni. La classificazione del testo è una classificazione intelligente del testo in categorie. Inoltre, l'utilizzo della machine learning per automatizzare queste attività rende l'intero processo estremamente veloce ed efficiente.

Dopo aver ampiamente analizzato tutte le varie tecniche disponibili allo stato dell'arte, è importante notare che, grazie all'evoluzione dell'informatica e alla continua condivisione di soluzioni comuni, allo stato della pratica, sono sempre maggiori le API di NLP messe a disposizione per risolvere i nostri scopi.

3.1 API NLP

L'acronimo API fa riferimento all'interfaccia di programmazione di un'applicazione e può assumere diverse forme. Si tratta, in linea generale, di un insieme di funzioni che permettono di accedere ai servizi di un'applicazione mediante un linguaggio di programmazione. Ma queste funzioni possono essere diverse: può trattarsi di una libreria che consente a uno sviluppatore di interagire con un determinato software o una piattaforma, oppure possono fungere da "chiamata" di un programma verso parti di un'altra applicazione.

Le API possono quindi essere definite come uno strumento flessibile ed efficiente sia per lo scambio di dati che per la condivisione di funzionalità: gli sviluppatori utilizzano questi strumenti per estendere o migliorare le funzionalità di un programma, una piattaforma, una soluzione software, attingendo a risorse di altri sistemi (altrimenti inaccessibili).

L'analisi dell'intento, delle emozioni e del sentimento dei dati testuali sono alcune delle parti più importanti della classificazione del testo.

Alcuni popolari piattaforme di servizio per gli scopi a noi necessari sono presentati in tabella.

È importante notare che tutte le API presentate, comprendono anche una fase di elaborazione del testo; quindi la fase di pre-elaborazione e di rappresentazione sono incluse al loro interno.

<i>API</i>	<i>FEATURE</i>	<i>FUNZIONALITA'</i>	<i>LICENZA</i>	<i>LINGUA</i>
<u>LUIS.ai</u>	Tutte le applicazioni LUIS sono incentrate su un argomento specifico del dominio o sui contenuti correlati. Apprendimento attivo. È possibile utilizzare modelli pre-esistenti, di livello mondiale e pre-costruiti da Bing e Cortana.	sentiment analysis, key phrase extraction, named entity recognition, language detection, Entity Linking	Gratis fino a 5000 transazioni	Olandese, inglese, francese, tedesco, italiano, polacco, portoghese, russo, spagnolo, svedese
<u>Wit.ai</u>		Entities, Intents, Context, Actions Natural Language Process (NLP)	Gratis	50 lingue tra cui, inglese, italiano, spagnolo, tedesco, russo
<u>Api.ai</u>	API.AI associa la query all'intento più adatto in base alle informazioni contenute nell'intento (esempi, entità utilizzate per annotazioni,	Intent, Entity, Context, Action, event, training	Gratis	Brazilian Portuguese, Chinese, English, Dutch,

	<p>contesti, parametri, eventi) e il modello di apprendimento automatico dell'agente.</p> <p>API.AI trasforma il testo della query in dati utilizzabili e restituisce i dati di output come oggetto risposta JSON.</p> <p>Sfrutta pacchetti di conoscenza predefiniti raccolti per diversi anni.</p>			French, German, Italian, Japanese, Portuguese, Russian, Spanish.
<u>Watson</u>	Costruito su una rete neurale, permette la creazione di chatbot	Intent, Entity and Dialog	Gratis, Standard, Premium	English, Japanese
<u>DeepPavlon</u>	<p>set di modelli NLP pre-formati, componenti del sistema di dialogo predefiniti (basati su ML / DL / regole) e modelli di pipeline;</p> <p>una struttura per implementare e testare i propri modelli di dialogo;</p> <p>strumenti per l'integrazione dell'applicazione con l'infrastruttura adiacente (messaggistica, software di helpdesk, ecc.);</p> <p>ambiente di benchmarking per modelli conversazionali e accesso uniforme a dataset rilevanti</p>	Data pre-processing, intent, NER, KBQA	Gratuita	Inglese e russo

<u>Dandelion</u>	<p>Dandelion API sfrutta il suo Knowledge Graph sottostante, senza fare affidamento sulle pipeline NLP tradizionali. Ciò lo rende più veloce, più scalabile, più facile da personalizzare e in modo indipendente dalla lingua.</p> <p>Le entità estratte sono collegate con l'enorme quantità di dati aggiuntivi disponibili nel nostro Knowledge Graph interno, che contiene dati aperti e proprietari di alta qualità. Questo Knowledge Graph è a sua volta collegato al Linked Open Data Cloud.</p> <p>Facile da integrare nei sistemi esistenti tramite una potente API REST, il motore gira su un'infrastruttura scalabile in grado di elaborare milioni di documenti al giorno.</p>	<p>Entity extraction, text similarity, text classification and sentiment analysis, wikisearch API.</p>	Gratuito fino a mille chiamate al giorno.	Tedesco, inglese, spagnolo, francese, italiano, russo
<u>BabeFly</u>	approccio unificato, multilingue, basato su grafici, al collegamento di entità e alla disambiguazione del senso di parola basato su una libera identificazione dei significati della parola	Entity linking and word sense disambiguation	Gratuita	271 lingue

	<p>associata con un'euristica del sottografo più densa che seleziona interpretazioni semantiche ad alta coerenza.</p> <p>Babelfy si basa sulla rete semantica multilingue e svolge congiuntamente disambiguazione e collegamento di entità.</p>			
<u>Core NLP []</u>	fornisce una serie di strumenti tecnologici per il linguaggio umano. Può dare le forme base delle parole, le loro parti del discorso, se sono nomi di aziende, persone, ecc., Normalizzare date, tempi e quantità numeriche, segnare la struttura delle frasi in termini di frasi e dipendenze sintattiche, indicare quali frasi nominali si riferiscono alle stesse entità, indicano sentiment, estrapolano relazioni particolari o open-class tra menzioni di entità, ottengono le citazioni di persone, ecc.	NER, Context, part-of-speech tagger, parser, sentiment analysys, bootstrapped pattern learning	Gratis	Inglese, arabo, cinese, tedesco, spagnolo.
<u>Tint [18]</u>	è una pipeline basata su Java per Natural Language Processing (NLP) in italiano.	Pre-processing text. POS tagging, NER, parsing, entity	Gratis	Italiano

è basato su Stanford CoreNLP e può essere utilizzato come strumento autonomo, incluso come libreria Java o come servizio API REST. Tint include anche wrapper (per strumenti di terze parti) che utilizzano il paradigma CoreNLP e quindi possono essere applicati a lingue diverse dall'italiano.

<u>Amazon Comprehend</u>	<p>è un servizio di elaborazione del linguaggio naturale che adotta l'apprendimento automatico per trovare informazioni e relazioni in un testo. Nessuna esperienza di apprendimento automatico richiesta.</p> <p>adotta l'apprendimento automatico per aiutare a estrarre informazioni e relazioni nei dati non strutturati. È inoltre possibile utilizzare le funzionalità AutoML disponibili in Amazon Comprehend per creare un insieme di entità personalizzate o modelli di classificazione di testo adattati unicamente alle esigenze della tua organizzazione.</p>	<p>Sentiment analysys, syntax analysys, NER, personalized entity, language detection, intent classification, topic modelling, multi-language support.</p>	<p>Gratis fino a 50000 unità di testo</p>	<p>Inglese, francese, tedesco, italiano, spagnolo</p>

3.2 LIBRERIE PYTHON DISPONIBILI

3.2.1 GENSIM

Gensim è una libreria open-source per la modellazione di argomenti senza supervisione e l'elaborazione del linguaggio naturale, utilizzando l'apprendimento automatico moderno delle macchine. Include streaming implementazioni parallelizzate di FastText, word2vec e doc2vec, nonché analisi semantica latente (LSA, LSI, SVD), matrice di fattorizzazione non negativa (NMF), tf -idf e proiezioni casuali. Inoltre grazie a Gensim è possibile caricare un modello pre-addestrato e utilizzarlo come modello per elaborare nuovi dati

3.2.2 NLTK

NLTK è una piattaforma per la creazione di programmi Python per lavorare con i dati del linguaggio umano. Fornisce interfacce di facile utilizzo a oltre 50 lingue e risorse lessicali come WordNet, insieme a una suite di librerie di elaborazione del testo per classificazione, tokenizzazione, derivazione, etichettatura, analisi e ragionamento semantico, e un forum di discussione attivo. Essa è molto impiegata in ambito didattico.

3.2.3 SPACY

SpaCy è una libreria open source per l'elaborazione del linguaggio naturale, scritta in Python e Cython. La libreria implemets modelli statistici di reti neurali in inglese, tedesco, spagnolo, portoghese, francese, italiano, olandese e greco; inoltre offre funzionalità di NER e di rimozione delle stop word, tokenization, POS tagging, lemmatization, stemming.

In particolare, è disponibile la libreria “it_core_news_sm” ovvero una rete neurale, nello specifico una CNN multi-task, in lingua italiana addestrata sulle dipendenze

universali e sul corpus WikiNER. Essa assegna vettori token specifici del contesto, tag POS, analisi delle dipendenze ed entità denominate. Supporta l'identificazione di entità PER, LOC, ORG e MISC.

3.2.4 SCIKIT-LEARN

Scikit-learn è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python. Essa contiene algoritmi di classificazione, regressione e clustering (raggruppamento) e macchine a vettori di supporto, regressione logistica, classificatore bayesiano, k-mean e DBSCAN, ed è progettato per operare con le librerie NumPy e SciPy.

Essa è composta da più librerie interne, suddivise in base allo scopo di lavoro:

1. Preprocessing: composto da diverse funzioni di utilità comune e classi di trasformatori per modificare i vettori delle feature non elaborate in una rappresentazione più adatta. Esso è risultato utile per più scopi come ad esempio l'encoding delle categorie con il “OneHotEncoder”, oppure il calcolo del tf-idf con il “TfidfVectorizer”;
2. Classificatori: scikit-learn include al suo interno molti classificatori come ridge regression, SVM, Stochastic Gradient Descent, Nearest Neighbors, alberi decisionali, e anche metodi misti. I modelli da noi utilizzati saranno, SVM e Naive Bayes.
3. Metriche: permette di effettuare calcoli di confronto, validazione e scelta di parametri e modelli. Esso implementa funzioni che valutano l'errore di previsione per scopi specifici. Inoltre permette anche di effettuare validazione attraverso cross-validation e ci permette quindi anche di suddividere nel giusto modo il dataset.

3.2.5 TENSORFLOW

L'apprendimento automatico può diventare complesso rapidamente e i modelli di apprendimento approfonditi possono diventare grandi. Per molti modelli di grafici, è necessario un allenamento distribuito per poter iterare entro un ragionevole lasso di tempo. In genere, i modelli che sviluppi devono essere distribuiti su più piattaforme.

Con la versione corrente di TensorFlow, si scrive codice per creare un grafico di calcolo, quindi eseguirlo. Il grafico è una struttura dati che descrive completamente il calcolo che si desidera eseguire.

TensorFlow è una libreria software open source per il calcolo numerico che utilizza i grafici del flusso di dati. È stato originariamente sviluppato da Google Brain Team all'interno dell'organizzazione di ricerca Machine Intelligence di Google per l'apprendimento automatico e la ricerca di reti neurali profonde, ma il sistema è abbastanza generale da essere applicabile anche in una vasta gamma di altri domini.

TensorFlow è multipiattaforma. Funziona su quasi tutto: GPU e CPU, comprese le piattaforme mobili e incorporate.

Il motore di esecuzione distribuito TensorFlow astrae i numerosi dispositivi supportati e fornisce un core ad alte prestazioni implementato in C++ per la piattaforma. TensorFlow. A parte questo, i frontend Python e C++.

L' API Layers fornisce un'interfaccia più semplice per i layer di uso comune nei modelli di deep learning. A ciò si aggiungono API di livello superiore, tra cui Keras e l'Estimator API, che facilita la formazione e la valutazione dei modelli distribuiti.

L'architettura di Tensorflow funziona in tre parti:

- Pre-elaborazione dei dati
- Costruisci il modello
- Allenare e stimare il modello

Si chiama Tensorflow perché prende l'input come un array multidimensionale, noto anche come tensori.

Un tensore è un **vettore** o una **matrice** di n-dimensioni che rappresenta tutti i tipi di dati. Tutti i valori in un tensore tengono identico tipo di dati con una nota (o parzialmente nota) **forma**. La forma dei dati è la dimensionalità della matrice o della matrice. Un tensore può essere originato dai dati di input o dal risultato di un calcolo. In TensorFlow, tutte le operazioni sono condotte all'interno di un **grafico** che si desidera eseguire su tale input. L'input viene inserito a un'estremità e quindi scorre attraverso questo sistema di più operazioni e esce dall'altra parte come output. Ogni operazione è denominata **nodo op** e sono collegate tra loro.

3.2.6 KERAS

Keras è una libreria open source per l'apprendimento automatico di reti neurali ad alto livello, scritta in Python. È progettata come un'interfaccia a un livello di astrazione superiore di altre librerie simili di più basso livello, e supporta come back-end le librerie TensorFlow, Microsoft Cognitive Toolkit (CNTK) e Theano. Progettata per permettere una rapida prototipazione di reti neurali profonde, si concentra sulla facilità d'uso, la modularità e l'estensibilità.

Un vantaggio nell'utilizzo di Keras è che permette l'utilizzo di una libreria di apprendimento approfondito che supporta reti convoluzionali, reti ricorrenti ed anche un mix di entrambe.

Oggi ci sono innumerevoli framework di deep learning disponibili, tra cui PyTorch e TensorFlow, ma nonostante ciò, Keras risulta uno dei framework più utilizzato proprio perché dà la priorità all'esperienza degli sviluppatori, infatti:

-Keras è un'API progettata per gli esseri umani, non per le macchine. Keras segue le best practice per ridurre il carico cognitivo : offre API coerenti e semplici, riduce al minimo il numero di azioni dell'utente richieste per casi d'uso comuni e fornisce un feedback chiaro e attuabile in caso di errore dell'utente.

-Questo rende Keras facile da imparare e facile da usare. Come utente di Keras, sei più produttivo, permettendoti di provare più idee della tua competizione, più velocemente il che a sua volta ti aiuta a vincere le gare di apprendimento automatico.

-Questa facilità d'uso non ha il costo di una flessibilità ridotta: poiché Keras si integra con i linguaggi di apprendimento approfondito di livello inferiore (in particolare TensorFlow), consente di implementare qualsiasi cosa si possa aver costruito nella lingua di base. In particolare, come tf.keras, l'API Keras si integra

perfettamente con i flussi di lavoro TensorFlow.

3.2.7 PYTORCH

PyTorch è un framework di deep learning, sviluppato principalmente dal Facebook AI Research (FAIR) group, che ha guadagnato una enorme popolarità fra gli sviluppatori grazie alla combinazione di semplicità ed efficienza. PyTorch ha un modo unico di costruire reti neurali: usare e riprodurre un registratore a nastro.

La maggior parte dei framework come TensorFlow, Theano, Caffe e CNTK hanno una visione statica del mondo. Bisogna costruire una rete neurale e riutilizzare la

stessa struttura ancora e ancora. Cambiare il comportamento della rete significa che bisogna ricominciare da capo. PyTorch utilizza una tecnica chiamata differenziazione automatica in modalità inversa, che consente di modificare il modo in cui la rete si comporta in modo arbitrario con zero lag o overhead.

3.2.8 HYPEROPT

Ogni rete neurale ha i propri iperparametri, come anche per SVM e Naive Bayes. Tali iperparametri sono stati semi-automaticamente calcolati da Hyperopt.

Hyperopt è un modo per cercare attraverso uno spazio iperparametrico. Hyperopt può utilizzare l' algoritmo Parzen Estimator (TPE) strutturato ad albero , che esplora in modo intelligente lo spazio di ricerca restringendo i parametri migliori stimati. È quindi un buon metodo per l'ottimizzazione di una rete neurale che utilizza metodi di discesa del gradiente e la regolazione degli iperparametri deve essere eseguita in modo diverso poiché la discesa del gradiente non può essere applicata. Pertanto, Hyperopt può essere utile non solo per sintonizzare iperparametri come il tasso di apprendimento, ma anche per regolare più parametri di fantasia in modo flessibile, come cambiare il numero di strati di certi tipi, o il numero di neuroni in un livello, o anche il tipo di livello da utilizzare in un determinato punto della rete, data una serie di scelte, ciascuna con iperparametri sintonizzabili annidati.

Esistono due principali metodi comuni di ottimizzazione dei parametri: ricerca della griglia e ricerca casuale. Questa è una ricerca casuale orientata, in contrasto con una ricerca di griglia in cui gli iperparametri sono prestabiliti con incrementi di passi fissi. La ricerca casuale dell'iper-parametrizzazione (come quella che fa Hyperopt) ha dimostrato di essere una tecnica di ricerca efficace.

Ognuno ha i suoi pro e contro. La ricerca della griglia è lenta ma efficace nella ricerca dell'intero spazio di ricerca, mentre la ricerca casuale è veloce, ma potrebbe perdere punti importanti nello spazio di ricerca.

Per riassumere, è più efficiente cercare casualmente attraverso i valori e restringere in modo intelligente lo spazio di ricerca piuttosto che eseguire il ciclo su set fissi di valori per gli iperparametri.

`oblib.dump()` e `joblib.load()` forniscono una sostituzione affinché pickle funzioni in modo efficiente su oggetti Python arbitrari contenenti dati di grandi dimensioni, in particolare matrici numpy di grandi dimensioni.

3.2.9 JOBLIB

Joblib è un set di strumenti per fornire pipeline leggere in Python con l'obiettivo di fornire strumenti per ottenere facilmente prestazioni e riproducibilità di una qualsiasi struttura Python, dal modello di una rete neurale che deve mantenere tutte le sue funzionalità e caratteristiche, ad un particolare oggetto che deve essere preservato nella sua intera forma.

Joblib è ottimizzato per essere veloce e robusto su dati di grandi dimensioni in particolare e ha ottimizzazioni specifiche per array intorpiditi.

Esso è molto fondamentale per due principali caratteristiche:

1. Evita di calcolare la stessa cosa due volte: il codice viene spesso rieseguito più volte, ad esempio durante la prototipazione lavori pesanti dal punto di vista computazionale (come nello sviluppo scientifico), ma sono soluzioni artigianali per alleviare questo problema soggetto a errori e spesso porta a risultati non riproducibili.

2. Persistenza su disco in modo trasparente: la persistenza efficiente di oggetti arbitrari contenenti dati di grandi dimensioni è difficile. L'uso del meccanismo di memorizzazione nella cache di joblib evita la persistenza scritta a mano e collega implicitamente il file su disco al contesto di esecuzione dell'oggetto Python originale. Di conseguenza, la persistenza di joblib è buona per riprendere lo stato di un'applicazione o un lavoro computazionale, ad esempio dopo un arresto anomalo.

Nello specifico vengono molto utilizzate due importanti funzioni di joblib:

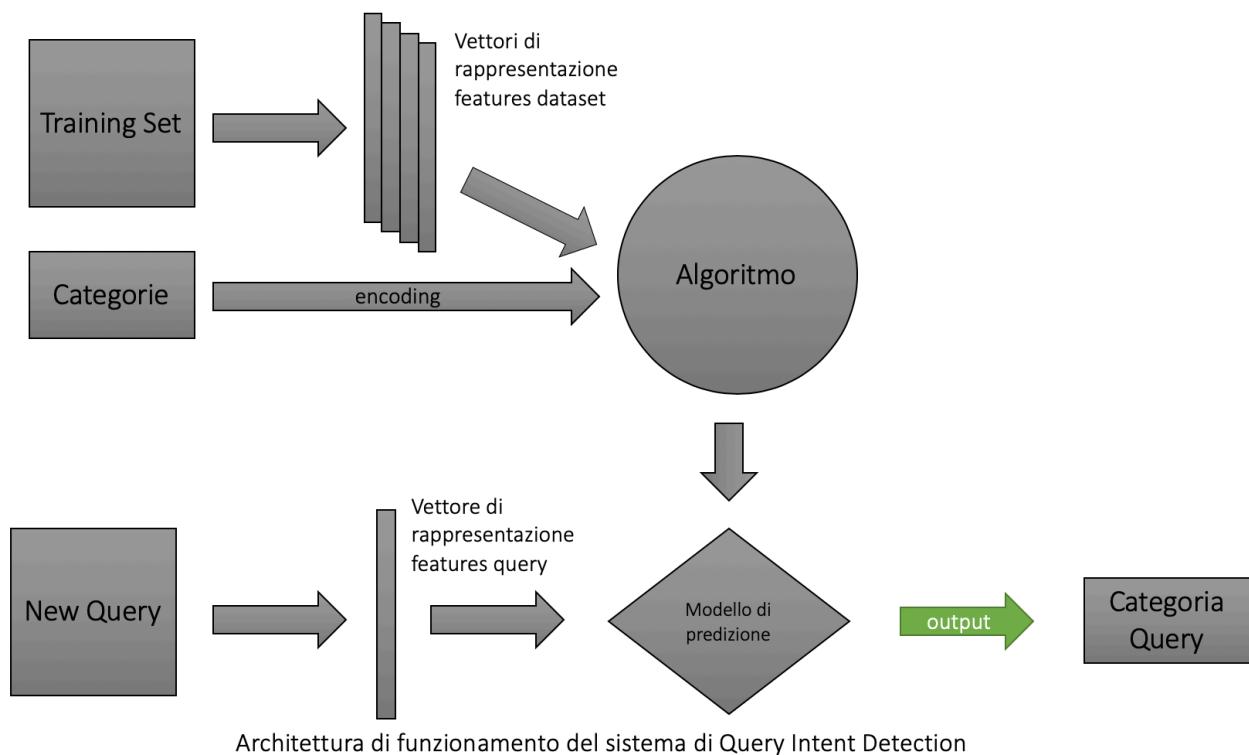
1. Dump: Persiste un oggetto Python arbitrario in un file.
2. Load: Ricostruisce un oggetto Python da un file persistente con joblib.dump.

joblib.dump () e joblib.load () forniscono una sostituzione affinché si possano conservare funzioni in modo efficiente su oggetti Python arbitrari contenenti dati di grandi dimensioni, in particolare matrici numpy di grandi dimensioni ad esempio o modelli complessi tratti fuori da una rete neurale.

CAPITOLO 4 - ARCHITETTURA E SVILUPPO

Nella progettazione e implementazione di tale sistema si è deciso prima di tutto di creare un algoritmo ad hoc che potesse lavorare bene nelle circostanze da noi richieste. Motivo per il quale si è deciso di non utilizzare le tecniche viste in API NLP a stato della pratica, ma di considerare solo l'apprendimento automatico supervisionato, focalizzandosi soprattutto sull'uso delle reti neurali. Si sono implementate diverse tecniche precedentemente presentate nello stato dell'Arte.

Il funzionamento generico che il sistema risultante, obiettivo di questa tesi, dovrà avere è descritto dalla seguente figura, in cui viene mostrata una architettura di funzionamento generica.



Come è possibile evidenziare dalla figura, il nostro sistema risulterà funzionare nel seguente modo: Per prima cosa bisogna addestrare un modello di predizione, creando un algoritmo che prendendo in pasto delle query già catalogate riesce a creare un modello appunto in grado di calcolare automaticamente, secondo delle distribuzioni probabilistiche ad esempio, e restituire come output, la categoria di una nuova query.

Una volta creato il modello, ogni nuova query verrà rappresentata nello stesso modo in cui è stato rappresentato il training set e subito dopo verrà dato in input al modello di predizione che sarà a questo punto in grado di determinare la categoria di appartenenza della nuova query inserita.

4.1 SCRAPING DATASET

A causa del fatto che il sistema è progettato principalmente in lingua italiana e a causa della scarsità di dataset contenenti query di carattere turistico, si è deciso sin dall'inizio di creare un dataset ad hoc che potesse al massimo soddisfare le richieste del progetto iniziale.

Per fare ciò si è cercato di determinare quali fossero le varie categorie di intent possibili da considerare data la query. Dopo una attenta analisi, esse risultano essere:

1. Luogo
2. Professione
3. Informazioni specifiche
4. Attività/Esperienze da svolgere

Identificate le classi di intent, è iniziata la fase di creazione del dataset resa possibile grazie alla creazione di procedure di scraping dal sito di QA Yahoo Answer uno dei motori di Query Answering più famoso al mondo.

Per fare ciò, si è creata una pagina html e php che permette, dato un URL (corrispondente ad una pagina di ricerca di Yahoo Answer) e una categoria di appartenenza delle domande contenute al suo interno, di salvare su un database l'elenco delle domande categorizzate con il corrispondente link.

I link dal quale estrarre le domande sono stati opportunamente scelti attraverso la ricerca di parole chiave che potessero indicare l'appartenenza ad una categoria

rispetto ad un'altra. Com'è possibile verificare nell'esempio riportato di seguito, con le parole chiave "cosa visitare" è stato possibile recuperare domande di vario tipo inerenti alla quarta categoria, ovvero Attività/Esperienze da fare.

The screenshot shows a search results page from Yahoo! Answers. The search bar at the top contains the query "cosa visitare". Below the search bar, there are several search results, each representing a question about things to visit:

- Vacanza a Madrid e Barcellona. Cosa visitare?**
per **visitare** Madrid si può iniziare dalla centralissima ... quale vengono calcolate le distanze di **Visitare** Tio Pepe Madrid tutte le autostrade che...
6 Answers · Viaggi · 21 lug 07:11
- TAORMINA e dintorni, cosa visitare?**
... dettagliate! Come prima **cosa** una bella passeggiata per ...panoramica che offre Taormina devi **visitare** La Villa Comunale di Taormina...
3 Answers · Viaggi · 05 giu 07:23
- Barcellona... cosa visitare?**
...ciao, ci sono tantissime **cose** da vedere dipende da quanto tempo hai.... Lassù trovi molte **cose** da vedere, innanzitutto tantissimo verde, ...
13 Answers · Viaggi · 01 apr 04:41
- Firenze per un fine settimana? Cosa visitare?**
dipende **cosa** ti piace! se ti piacciono... giornata per la fila. Per **visitari** occorrono almeno 3 ore. Se sei ...
4 Answers · Viaggi · 03 nov 22:35
- Viaggio in Irlanda, cosa visitare?**
Ciao, per **visitare** la vera verde Irlanda secondo me ...raggiungere l'Ulster (usano ancora la sterlina) e **visitare** Londonderry, le giant's causeway e Belfast...
3 Answers · Viaggi · 10 mar 13:23
- Viaggio ad Orlando cosa visitare?**
... persone si recano qui magari dall'Europa per **visitare** il parco, che si sviluppa in un'area immensa che s...
7 Answers · Viaggi · 18 ott 10:33
- Piccolo weekend a roma..cosa visitare?**
Cosa Visitare a Roma Ogni angolo della città è ricco...
6 Answers · Viaggi · 02 dic 13:42

Una volta identificato il link, esso viene inserito all'interno del seguente form con la corrispondente categoria, indicata numericamente.

https://it.answers.search
4
SCRAPE

All'interno del database verranno memorizzati nel seguente modo:

	id [PK] integer	domanda character varying (100)	url character varying (100)	categoria integer
11	952	Cosa mi consigliate di vede...	https://it.answers.yahoo.co...	4
12	958	luoghi di interesse artistico ...	https://it.answers.yahoo.co...	4
13	301	Cosa mi consigliereste di v...	https://it.answers.yahoo.co...	4
14	305	Visitare la casa di Maria da ...	https://it.answers.yahoo.co...	4
15	307	Consigli per visitare Napoli?	https://it.answers.yahoo.co...	4
16	333	dove possiamo andare dom...	https://it.answers.yahoo.co...	4
17	335	Week-end a Roma: Cosa mi ...	https://it.answers.yahoo.co...	4
18	339	week-end in puglia....cosa ...	https://it.answers.yahoo.co...	4
19	350	vacanza con tappe in italia, ...	https://it.answers.yahoo.co...	4

Si è in totale effettuato scraping di circa 2000 domande, cercando di considerare almeno 500 esempi per ogni categoria. È importante notare lo sforzo implementativo in tale fase, in quanto si sono dovuti cercare i giusti URL contenenti le domande corrispondenti alle 4 categorie sopra elencate. Inoltre, segue una fase di controllo di ogni domanda per verificarne la loro validità, in quanto Yahoo Answer presenta molti altri tipi di intent che non verranno affrontati dal sistema o domande poco inerenti o ambigue. Per evitare il bias umano, le domande sono state valutate da più persone.

Il dataset iniziale è stato suddiviso in due principali sottoinsiemi:

- training set: usato unicamente per addestrare la rete (cambiare i pesi), solitamente è il 80 % del set iniziale;
 - validation set: sottoinsieme del training set, generalmente il 15/20% del training set. Viene usato sia per testare la generalizzazione sia per settare gli iperparametri;
 - estimation set: restante dell'insieme usato per il training del modello.

- o test set: tutto ciò che non fa parte del training set (generalmente possiamo dimenticarcene, lo useremo solo per valutare il funzionamento su esempi mai visti).

4.2 PRE-ELABORAZIONE DEL TESTO

Una volta creato il dataset e installati i Tool necessari precedentemente descritti, si è proceduti con la prima importantissima fase, come già accennato nello stato dell'arte, ovvero quella di pre-elaborazione del testo, in cui si sono eliminate tutte le cosiddette “stop-word”, si sono portate inizialmente tutte le parole al lemma originale, subito dopo si è optato per una seconda strada in cui sono state sostituite in lemma solo tutte le parole troncate o con accenti come ad esempio “è” in “essere” e “po’” in “poco” e infine, in entrambi i casi, ogni query è stata suddivisa in token.

Si è deciso di utilizzare spaCy per la fase di pre-processing, poiché più completa rispetto a NTLK. Grazie a spaCy è stato possibile recuperare per ogni frase anche il PoS (Part of Speech)

4.3 WORD REPRESENTATION

Una volta pulito il testo ed effettuata la sua tokenizzazione si è deciso come effettuare la sua rappresentazione numerica. Si sono utilizzati più tecniche a partire da due modelli pre-addestrati e da uno, generato attraverso il calcolo del tf-idf.

Il primo modello utilizza il tool word2vec con approccio Skip-gram di dimensione 300, addestrato su un corpus di testo estratto da Wikipedia in lingua italiana creato dall'Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", Pisa. Esso produce un vocabolario di 733.392 parole diverse e ad ognuna di esse associa una similarità da se a 300 diverse parole. Nonostante la fase di pulizia del testo, alcune delle parole contenute nel nostro training Set non trovavano occorrenza nel vocabolario di questo

modello. Per far fronte a tale problema si è utilizzata una funzione random che, ogni volta che una parola non viene trovata, assegna ad essa la rappresentazione di una diversa parola casuale all'interno del vocabolario.

```
domanda = cosa posso visitare stasera a polignano??
[('cosa', 'PRON')]
[('posso', 'AUX')]
[('visitare', 'VERB')]
[('stasera', 'ADV')]
[('a', 'ADP')]
[('polignano', 'ADJ')]
[('?', 'PUNCT')]
[('?', 'PUNCT')]
I token trovati sono: 8
I token non trovati sono: 0
Il vocabolario è: 733392
le domande trovate sono: 1
(1, 50, 300)
```

Il secondo modello è stato testato anche per evitare di assegnare una rappresentazione casuale ad una parola non presente nel vocabolario. Esso utilizza il tool fastText ed è pre-addestrato su una scansione generica del web e su Wikipedia. Esso è stato addestrato usando CBOW con pesi di posizione, nella dimensione 300, con caratteri n-grammi di lunghezza 5, una finestra di dimensioni 5 e 10 negativi, creato in multilingua direttamente dagli ideatori di fastText. In tal caso il numero di vocaboli generati è di 2.000.000.

```
domanda = cosa posso visitare stasera a polignano??
[('cosa', 'PRON')]
[('posso', 'AUX')]
[('visitare', 'VERB')]
[('stasera', 'ADV')]
[('a', 'ADP')]
[('polignano', 'ADJ')]
[('?', 'PUNCT')]
[('?', 'PUNCT')]
I token trovati sono: 8
I token non trovati sono: 0
Il vocabolario è: 2000000
le domande trovate sono: 1
(1, 50, 300)
```

Il terzo modello è stato creato a partire dal training Set basato sul calcolo del tf-idf ovvero Term Frequency e Inverse Document Frequency. Inoltre, per tale tecnica si è deciso di rappresentare le parole con n-gram di parole uguali a due, ovvero le parole sono rappresentate a coppie, così da preservare il contesto di utilizzo di ogni parola, ad esempio “Cosa posso visitare stasera a Polignano?” sarà rappresentata come segue: “cosa posso”, “posso visitare”, “visitare stasera”, “stasera a”, “a Polignano”, “Polignano?”.

In fase di progettazione, si è provato a sostituire tutte le parole con i propri lemmi, ma visti gli scarsi risultati, si è deciso di procedere con la lemmatizzazione solo per le parole troncate come ad esempio “po” oppure “è” in “poco” ed “essere” rispettivamente.

Grazie all’utilizzo di joblib, è stato possibile salvare la matrice di word embedding creati e la matrice del calcolo con TF-IDF. Il loro salvataggio risulta molto utile per non rieseguire la Word Representation ad ogni tentativo di addestramento.

4.4 INTENT DETECTION

Una volta eseguita la rappresentazione delle query si è proceduto allo sviluppo dell’algoritmo per la creazione del modello di predizione. Si sono creati sei diversi modelli: LSTM, C-LSTM, BiLSTM, C-BiLSTM, Naive Bayes, SVM (Support Vector Machine).

Le reti neurali progettate sono state addestrate sul training test con l’utilizzo di entrambi i word embedding creati e successivamente con la rappresentazione calcolata con TF-IDF, si sono valutate le prestazioni del Naive Bayes ed SVM.

Per il nostro scopo si è utilizzato hyperopt per reti. Per le reti neurali è stata adottata una versione di back propagation ad epocha ovvero Batch Mode, con un numero di Batch pari a 8.

Tutte le reti neurali progettate e presentate, sono state create grazie all'utilizzo di Keras, utilizzando funzioni e parametri.

Il primo modello testato è stato LSTM, creata secondo criteri scelti da hyperopt; Essa risulta essere composta da 300 celle di memoria, dove ognuna di esse elabora e passa informazioni alla cella successiva, informazioni circa lo stato della cella e lo stato nascosto, come già spiegato nello stato dell'arte. Come funzione di attivazione si è utilizzata "Relu". Tale LSTM viene inoltre seguita da un livello di 200 neuroni Dense, seguito anch'esso da una funzione di attivazione Relu ed infine seguito da un ultimo strato di neuroni Dense, in questo caso solo 4 e con attivazione softmax, per essere in grado di trarre la soluzione in una delle 4 categorie possibili. In fine, si è applicata una ottimizzazione per la back propagation utilizzando Adam come algoritmo. I risultati di accuratezza ottenuti con questo modello, testandolo sulla porzione di validation, non sono stati molto soddisfacenti, infatti oscillavano fra 0.3 e 0.4.

Il secondo modello testato rappresenta una piccola evoluzione rispetto alla precedente, infatti si è progettata una BiLSTM. Tale BiLSTM risulta essere inizialmente formata da 200 celle di memoria che lavorano però in entrambi i lati così da riuscire a considerare anche il contesto a destra, quindi alla fine della frase. Tale livello è seguito da una funzione di attivazione Relu e da un livello Global Max Pooling in grado di interfacciarsi tra la BiLSTM e il layer Dense successivo ed evitare quindi l'overfitting. Infine, un layer Dense di dimensione 4 con attivazione softmax è posto alla fine del modello in modo tale da estrapolare le informazioni utili e determinare quale sia la categoria di appartenenza. Rispetto alla precedente rete, ovvero la semplice LSTM, essa risulta essere molto più performante, infatti oscilla tra un'accuratezza di 0.7 e 0.8, quasi il doppio. Merito sia della doppia direzione di ricerca che del livello di pooling.

Il terzo modello testato include la rete CNN nei modelli precedenti. Quindi si è progettata creando inizialmente una CNN con un numero di filtri di output dopo la convoluzione pari a 64, con una finestra di dimensioni pari a 5 e una funzione di attivazione “Relu” sui risultati di output. Finita la fase di convoluzione, si passa attraverso un Dropout di 0.2 e un è Max Pooling in grado di evitare il sovraccarico e intercorrersi fra le due strutture di reti neurali. Infatti, subito dopo, troviamo la stessa architettura della LSTM utilizzata singolarmente come primo modello. I pesi, anche in questo modello vengono propagate con un learning rate pari a 0,002 con l'algoritmo adam. Tale soluzione è risultata quasi pari alla precedente, nonostante il passaggio di convoluzione.

Infine, parlando di reti neurali, si è testato un modello che inglobasse quelli visti in precedenza ovvero C-BiLSTM una CNN, seguita da una LSTM bidirezionale.

C-BiLSTM passa prima attraverso il convolution layer che viene applicato sulla matrice delle parole della query convolvendo un filtro di dimensione 250, attraverso una finestra di dimensione 4, calcolati appositamente da Hyperopt, seguito da un layer di dropout 0.6, un max-pooling layer, per evitare l'overfitting.

A loro volta questi dati predetti vengono passati in input ad un altro hidden layer BiLSTM di dimensione 200 che codifica varie informazioni sequenziali nella query.

A questo punto viene applicato un Global max-pooling layer seguito da un ultimo hidden layer, Dense 300, sempre gestito da Hyperopt, per apprendere varie importanti features locali.

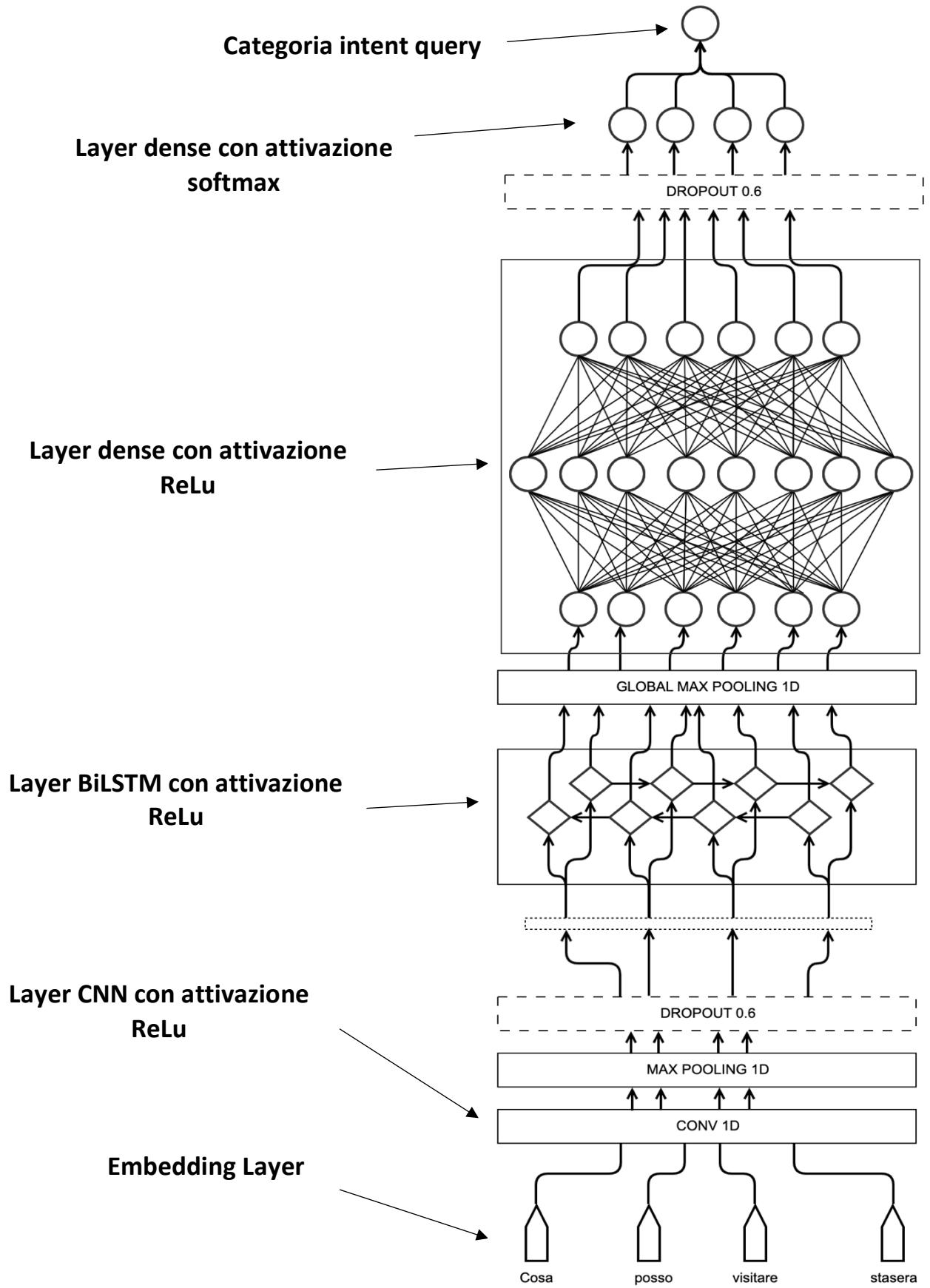
Dopo un layer di regolarizzazione Dropout 0.6, troviamo il layer Dense finale, di dimensione 4, con funzione di attivazione softmax, che modella le varie interazioni tra le feature e infine restituisce le probabilità di appartenenza della query alle 4 categorie secondo una distribuzione calcolata dalla rete neurale.

Per migliorare l'apprendimento ed evitare il sovraddattamento viene applicata una trasformazione lineare non elementare ad ogni output prodotto dagli hidden layer della rete neurale attraverso la Rectified Linear Unit (ReLU) nel layer di attivazione. Infine, ad ogni epoca, si effettua una ottimizzazione adam con un learning rate di 0.00249 per ottimizzare e per effettuare una back propagation con la sostituzione ad ogni ciclo dei pesi associati.

Una volta testate le varie reti neurali viste nello stato dell'arte, si è pensato di provare un modello Naive Bayes, che quindi calcola le probabilità che ogni parola o, come nel nostro caso, coppie di parole possano appartenere ad una categoria, piuttosto che ad un'altra. Si è scelto di creare tale modello facendo uso della libreria Scikit-Learn poiché risultava essere la più completa e la più usata. Per creare tale modello si è semplicemente effettuato il fit sui dati di training, senza specificare alcun parametro.

Infine, si è testato uno dei modelli che a stato dell'arte risulta quasi sempre essere il migliore. Tale modello prende il nome di SVM ovvero Support Vector Machine. Come spiegato nello stato dell'arte, essa usa le funzioni degli input originali come input della funzione lineare. Queste funzioni sono chiamate funzioni del kernel, che non sono altro che la codifica del prodotto scalare nello spazio dei positivi. Vengono utilizzate molte funzioni del kernel diverse. Una funzione del kernel di esempio è il prodotto delle funzionalità originali. Un SVM costruisce una superficie decisionale, che cioè un iperpiano che divide gli esempi positivi e negativi in un maggior numero di dimensioni. Aggiungere i prodotti delle funzionalità è sufficiente per abilitare la rappresentazione dell'esclusiva o della funzione. Aumentare la dimensionalità può, tuttavia, causare un sovradattamento. Questo viene evitato poiché i vettori di supporto definiscono una superficie che dipende da meno parametri rispetto al numero di esempi.

La struttura della rete neurale, costruita e infine adottata in questa tesi, è mostrata di seguito:



CAPITOLO 5 – VALUTAZIONE

La valutazione delle prestazioni di un modello è una delle fasi principali nel processo di analisi scientifica dei dati. Indica quanto è stato positivo il punteggio (stime) di un set di dati da un modello sottoposto a training. La valutazione e la convalida incrociata sono due modi standard di misurare le prestazioni del proprio modello. Entrambi generano metriche di valutazione che l'utente può usare per controllare o mettere a confronto quelle di altri modelli.

Nel mondo del machine learning, per valutare diversi modelli, è necessario fare affidamento ad una matrice di confusione o confusion matrix, anche conosciuta come matrice di errore. Essa rappresenta una specifica tabella che mostra le performance di un algoritmo di apprendimento supervisionato. Ogni riga di questa tabella rappresenta istanze di classi predette, le colonne invece le istanze di classi reali.

		True condition	
		Total population	
Predicted condition	Total population	Condition positive	Condition negative
	Predicted condition positive	True positive	False positive, Type I error
Predicted condition negative		False negative, Type II error	True negative

Essa risulta essere una tabella di contingenza speciale con due semplici dimensioni “reale” e “predetto” e le stesse istanze di classe in entrambe le dimensioni.

Una tabella di confusione è una tabella con 2 righe e due colonne che riportano il numero di falsi positivi (FP), falsi negativi (FN), veri positivi (TP), veri negativi (TN).

Come è possibile evidenziare dalla figura, esistono due tipi di errore: il primo indica il

false positive, ovvero quando la condizione è negativa in realtà e la predizione risulta essere positiva; il secondo indica il false negative, che, al contrario del precedente indica una situazione in cui in realtà la condizione è positiva ma il sistema predice negativamente.

Dato un test set è possibile enumerare quanti casi per ogni tipologia di situazione si verificano. Grazie a questo, è possibile calcolare molte metriche di valutazione come: precisione, richiamo, ed accuratezza.

5.1 METRICHE DI VALUTAZIONE

La precisione in un modello indica la correttezza ovvero la proporzione di esempi predetti come “positive” che sono effettivamente positivi, ed è calcolata nel seguente modo:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Il richiamo in un modello indica la completezza ovvero la proporzione di esempi effettivamente predetti che sono predetti come positivi, ed è calcolata nel seguente modo:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

La misurazione F1 è una misura dell'accuratezza di un test che considera sia la precisione p che il richiamo r del test per calcolare il punteggio. Essa rappresenta la media armonica del richiamo e della precisione calcolata nel seguente modo:

$$F1 = 2 \times \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

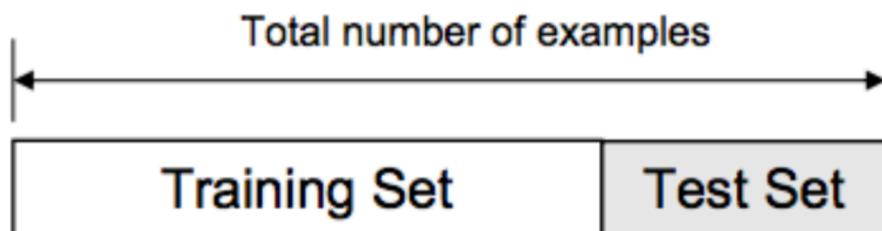
Quest'ultima ci restituisce nel migliore dei casi 1 (precisione e richiamo perfetti) nel peggiore 0 (viceversa).

5.2 TRAINING/TEST SPLIT VALIDATION

Il dataset utilizzato è stato suddiviso in dati di allenamento e dati di test. Il set di training contiene un output noto e il modello apprende su questi dati per essere successivamente generalizzato ad altri dati. Abbiamo il set di dati di prova (o sottoinsieme) per testare la previsione del nostro modello su questo sottoinsieme.

Più nello specifico si è suddiviso in:

- training set: usato unicamente per addestrare la rete (cambiare i pesi), solitamente è il 80 % del set iniziale;
 - validation set: sottoinsieme del training set, generalmente il 15/20% del training set. Viene usato sia per testare la generalizzazione sia per settare gli iperparametri;
 - estimation set: restante dell'insieme usato per il training del modello in fase di addestramento.
- test set: tutto ciò che non fa parte del training set (generalmente possiamo dimenticarcene, lo useremo solo per valutare il funzionamento su esempi mai visti).

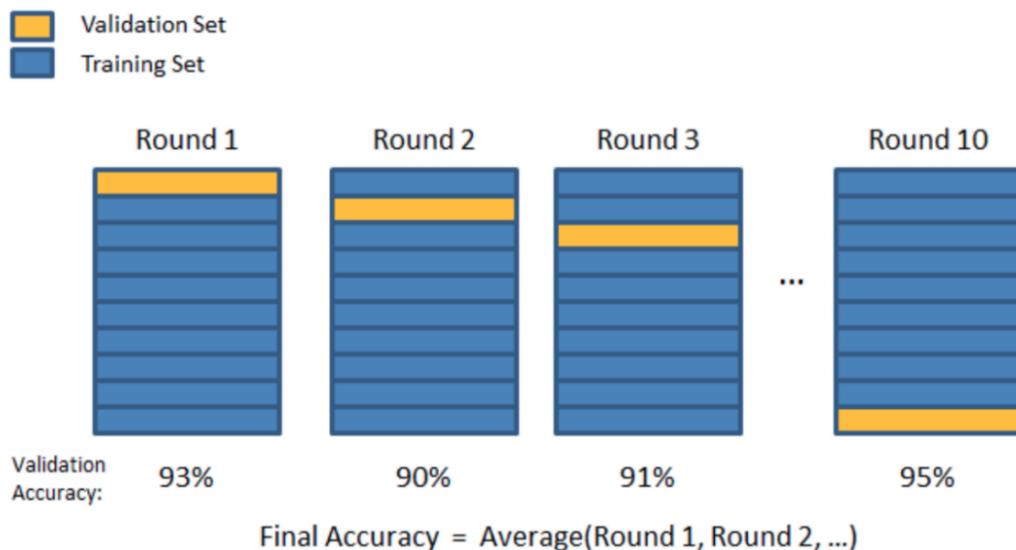


Per la fase di valutazione finale si utilizzerà quindi la porzione test set, utilizzando le metriche di valutazione nel paragrafo precedente valutate.

5.2 CROSS-VALIDATION

Quest'ultima tecnica di valutazione utilizzata molte volte effettua una verifica solo su una parte (testingSet) del dataset che risulta essere standard, ovvero la porzione di dati che viene presa non casualmente e che magari potrebbe estrarre solo dati con determinate caratteristiche. Ciò si traduce in sovraccarico, anche se è proprio quello che stiamo cercando di evitare. È qui che entra in gioco la validazione incrociata ovvero **cross-validation**.

L'idea principale alla base della cross-validation è che ogni campione nel nostro set di dati ha l'opportunità di essere testato. La figura mostrata di seguito indica come, la convalida incrociata riesce a valutare tutte le porzioni di esempi, in 10 round.



Il modello viene addestrato su ogni sottoinsieme ad eccezione di uno che sarà proprio il validation set. Questa procedura viene ripetuta K volte utilizzando per il validation set a turno uno dei K sottoinsiemi. Lo svantaggio di questa tecnica è che richiede molta computazione dal momento che il modello deve essere addestrato K volte.

Questo produce k modelli addestrati: ognuno di essi risulta addestrato su un set di dati parzialmente sovrapposti e viene valutato su un set di dati non sovrapposti. In fine per ogni modello viene calcolata l'accuratezza, di solito con la misurazione F1.

5.3 McNEMAR TEST

Nel suo importante e ampiamente citato articolo del 1998 sull'uso di test di ipotesi statistiche per confrontare i classificatori intitolato “Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms”, [27], Thomas Dietterich raccomanda l'uso del test di McNemar per la valutazione di due modelli di apprendimento supervisionato.

In particolare, il test è raccomandato nei casi in cui gli algoritmi che vengono confrontati possono essere valutati una sola volta, ad esempio su un set di test, al contrario di valutazioni ripetute tramite una tecnica di ricampionamento, come la validazione incrociata di k-fold.

I test statistici che possono confrontare modelli basati su un singolo set di test sono una considerazione importante per l'apprendimento automatico moderno, in particolare nel campo dell'apprendimento profondo. Il test di McNemar può essere un test adatto per valutare questi modelli di apprendimento profondo di grandi dimensioni e ad addestramento lento.

Come spiegato approfonditamente da Sebastian Raschka in [27], il test di McNemar è un test statistico non parametrico per confronti accoppiati che può essere applicato per confrontare le prestazioni di due classificatori di apprendimento automatico.

Spesso, il test di McNemar, chiamato anche "test chi-quadrato con soggetti", viene applicato a dati nominali accoppiati basati su una versione di matrice di confusione 2x2 (a volte anche denominata tabella di contingenza 2x2) che confronta le previsioni di due modelli tra loro (da non confondere con le tipiche matrici di confusione riscontrate nell'apprendimento automatico, che elencano i conteggi falsi positivi, veri positivi, falsi

negativi e veri negativi di un singolo modello). Il layout della matrice di confusione 2x2 adatta per il test di McNemar è mostrato nella figura seguente:

		Model 2 correct	Model 2 wrong
		A	B
Model 1 correct	A		
	C	D	

Data una tale matrice di confusione 2x2, possiamo calcolare l'accuratezza di un Modello 1 tramite $(A + B) / (A + B + C + D)$, dove $A + B + C + D$ è il numero totale di esempi di test n.

Allo stesso modo, possiamo calcolare l'accuratezza del Modello 2 come $(A + C) / n$.

I numeri più interessanti in questa tabella sono nelle celle B e C, poiché contano semplicemente il numero di campioni in cui sia il Modello 1 che il Modello 2 hanno fatto previsioni corrette, mentre A e D rispettivamente, calcolano le previsioni errate.

Le celle B e C (le voci fuori diagonale), ci dicono in che modo differiscono i modelli.

Nel Test di McNemar, formuliamo l'ipotesi nulla che le probabilità p (B) e p (C) (dove B e C si riferiscono alle celle della matrice di confusione introdotte in una figura precedente) sono le stesse, o in termini semplificati: nessuna delle due modelli si comportano meglio dell'altro. Pertanto, potremmo considerare l'ipotesi alternativa che le prestazioni dei due modelli non sono uguali.

La statistica del test McNemar ("chi-quadrato") può essere calcolata come segue

$$\chi^2 = \frac{(B - C)^2}{B + C}$$

Dopo aver impostato una soglia di significatività, ad esempio $\alpha = 0,05$, possiamo calcolare un valore p; supponendo che l'ipotesi nulla sia vera, il valore p è la probabilità di osservare il dato valore empirico (o più grande) χ^2 . Se il valore p è inferiore al livello di significatività scelto, possiamo respingere l'ipotesi nulla che le prestazioni dei due modelli siano uguali, viceversa il contrario, ovvero più il p value è sotto il nostro livello di significatività, più saranno concordi i due modelli.

5.4 RISULTATI OTTENUTI

Inizialmente, viene considerato il set di dati già diviso come trainingSet e testingSet. Il training set è ulteriormente ripartito in due insiemi disgiunti per validare il modello su un set di dati diverso da quello usato per l'addestramento: si valida sul validation set e non sull'estimation set. In questo modo possiamo usare il training set per stimare le prestazioni dei vari modelli candidati e scegliere il migliore. C'è comunque la possibilità che il modello più performante in realtà è incappato in un overfitting del validation set. È possibile ricorrere ad alcune tecniche particolari di test effettuate sull'intero dataset per verificare la capacità di generalizzazione della rete. Queste tecniche prendono il nome di cross validation, la più comune è la k-fold cross-validation che divide il set di N esempi in k sottoinsiemi

Una volta creati tutti i modelli di predizione per ogni diverso algoritmo, si è testato il loro funzionamento andando a somministrare loro il Testing Set, ovvero quella porzione di domande sconosciute ai modelli. Ogni modello ha dato i propri risultati e grazie all'utilizzo di Scikit-Learn si sono potuti calcolare i valori di accuratezza, precisione, richiamo e F1, ottenendo i seguenti risultati:

	model	accuracy	precision	recall	f1
0	w2v LSTM	0.392241	0.196154	0.372535	0.256431
1	w2v C-LSTM	0.392241	0.196154	0.372535	0.256431
2	w2v nolem LSTM	0.288793	0.072198	0.250000	0.112040
3	w2v nolem BiLSTM	0.758621	0.742565	0.749284	0.743008
4	w2v nolem C-LSTM	0.745690	0.743718	0.747266	0.733677
5	w2v nolem C-BiLSTM	0.788793	0.776099	0.784633	0.775471
6	FT nolem LSTM	0.288793	0.072198	0.250000	0.112040
7	FT nolem BiLSTM	0.737069	0.717138	0.722366	0.719002
8	FT nolem C-LSTM	0.711207	0.705855	0.719506	0.704721
9	FT nolem C-BiLSTM	0.750000	0.738093	0.758173	0.741520
10	NAIVE	0.637931	0.725560	0.646121	0.637289
11	SVM	0.793103	0.781413	0.781108	0.779122

Com'è possibile notare dalla tabella SVM risulta avere una precisione ed una accuratezza maggiore rispetto a tutti i modelli e di poco superiore rispetto alla C-BiLSTM con rappresentazione word2vec, ma al contrario, il maggior richiamo è dato da quest'ultima rete neurale. La differenza fra questi ultimi due modelli nella misurazione F1 è quasi impercettibile.

Si è deciso di effettuare cross-validation solo per i 2 modelli migliori, ovvero SVM e C-BiLSTM, con un numero di fold k pari a 10 e si è calcolata, grazie a scikit-learn è stato possibile, ad ogni fold, calcolare precisione, richiamo e F1. Si riportano nella seguente tabella i risultati di F1 di entrambi i modelli:

F1 MEASURE	C-BiLSTM	SVM
K = 1	0.985	0.817
K = 2	0.992	0.816
K = 3	1.0	0.838
K = 4	0.990	0.794
K = 5	0.989	0.782
K = 6	0.985	0.783

K = 7	1.0	0.796
K = 8	1.0	0.856
K = 9	0.980	0.835
K = 10	0.992	0.850

Come è possibile evidenziare dalla tabella, la rete neurale si comporta molto bene nella convalida incrociata a differenza di SVM, che mantiene in ogni caso dei buoni valori ma inferiori alla rete neurale.

In fine, si sono testati i vari modelli tra di loro con il test statistico di significatività di McNemar.

Il test di McNemar è stato implementato usando la funzione Statsmodels mcnemar (). La funzione accetta la tabella di contingenza come argomento e restituisce la statistica di prova calcolata e il valore p. I risultati così ottenuti sono riportati nella tabella di seguito.

X	W2VNLBiLSTM	W2VNLCLSTM	W2VNLCBiLSTM	FTNLBiLSTM	FTNLCLSTM	FTNLCBiLSTM	SVM
W2VNLBiLSTM	0.0	0.728	0.21	0.458	0.126	0.868	0.243
W2VNLCLSTM	0.728	0.0	0.076	0.868	0.268	0.005	0.144
W2VNLCBiLSTM	0.21	0.076	0.0	0.058	0.005	0.122	1.0
FTNLBiLSTM	0.458	0.868	0.058	0.0	0.392	0.69	0.047
FTNLCLSTM	0.126	0.268	0.005	0.392	0.0	0.078	0.005
FTNLCBiLSTM	0.868	0.005	0.122	0.69	0.078	0.0	0.132
SVM	0.243	0.144	0.047	0.047	0.005	0.132	0.0

Come è possibile notare dalla tabella, sono pochi i casi in cui è possibile accettare l'ipotesi nulla. Risalta subito però, che il p-value corrispondente al confronto fra il modello C-BiLSTM con rappresentazione vettoriale prodotta attraverso word2vec, e il modello SVM risulta essere uguale a 1. Questo sta a significare che è possibile accettare l'ipotesi nulla che i due modelli sono uguali e, ancora più specificatamente, indica che i due modelli sono completamente d'accordo su ogni predizione.

CAPITOLO 6 – CONCLUSIONI E SVILUPPI FUTURI

6.1 CONCLUSIONI

Il progetto FEEL@HOME ha come obiettivo quello di costruire una piattaforma sociale e tecnologica che faccia sentire ogni ospite (turista, ...) a casa, ovvero una piattaforma che si ponga come obiettivo, il contatto, l'esperienza, l'incontro fra i locali (residenti) e i numerosi ospiti che visitano il nostro territorio.

L'ospite spesso si affida a consigli raccolti sul web, per far fronte ai suoi bisogni. A tal proposito, il progetto “Feel at Home” persegue l’obiettivo di realizzare un “ecosistema di prossimità” utilizzabile via dispositivo mobile che favorisce contatti mirati fra ospiti e persone del posto (locali). Lo scambio di nozioni, informazioni e consigli diventa il punto focale della piattaforma che permette di supportare le necessità dell'ospite valorizzando i locali che decidono di mettersi a disposizione per far vivere a pieno il territorio.

In questo tesi si è dimostrato che determinare l'intento dietro la query dell'utente ha il potenziale per migliorare notevolmente le prestazioni di un motore di ricerca. L'intento della query dell'utente influenza tutti i processi decisionali nel progetto FEEL@HOME. Diversi tipi di query, ovvero diversi tipi d'intenzione, hanno implicazioni diverse sulla tipologia di persona con il quale vorrà comunicare l'utente.

Dopo aver valutato i vari modelli proposti con le metriche nel capitolo precedente presentate, è possibile tirare delle conclusioni.

È sorprendente notare come gli approcci più tradizionali come Naive Bayes e SVM hanno fornito risultati incoraggianti. Tuttavia, le reti neurali hanno mostrato un comportamento quasi inferiore alle tecniche di cui sopra, ad eccezione di quelle più elaborate, come C-LSTM e la C-BiLSTM. La causa di tali risultati è riconducibile al contenuto numero di esempi di training con i quali si sono addestrate le reti. In termini

di rappresentazione vettoriale delle parole, gli approcci più tradizionalmente utilizzati, ovvero il TF-IDF con sublinear tf-scaling, risulta meno efficace rispetto a rappresentazioni sotto forma di embedding ad alta dimensione, che sono in grado di incorporare significati semantici nelle rappresentazioni vettoriali. Tra i due embedding valutati, nell'implementazione finale, si è deciso di utilizzare quello basato su word2vec, poiché mostrava risultati migliori, a parità di modelli di rete.

Nella convalida incrociata, dove si è deciso di valutare solo i due modelli con F1 measure maggiore, ovvero C-BiLSTM e SVM, è stato possibile notare come la rete neurale riesca a imparare più facilmente e quindi a produrre risultati di F1 superiori rispetto a SVM.

Per una valutazione più accurata si è deciso di valutare i modelli prodotti andando ad identificare le differenze significative tra le predizioni calcolate a partire dagli stessi dati. Grazie al McNemar test, è stato possibile evidenziare una differenza poco significativa tra SVM e C-BiLSTM, in particolare, che quindi indicano che entrambi i modelli sono quasi completamente d'accordo per ogni predizione.

Pertanto, si è deciso di implementare la rete neurale C-BiLSTM ovvero una combinazione di reti basate su reti convoluzionali, reti ricorrenti, utilizzate sia per la fase di word representation che per la fase di intent detection.

Tale scelta è determinata sia dalla F1 Measure superiore ottenuta dalla rete neurale nella convalida incrociata e sia perché, supponendo sviluppi futuri, si potrebbe implementare un sistema di ricalcolo del modello di predizione ad ogni aggiunta di nuova query (magari da parte degli utenti finali in fase di utilizzo del sistema). In questo modo la rete neurale, sviluppa un meccanismo di apprendimento che consentirà una specializzazione sempre più accurata (Reinforcement Learning), in grado di ampliare la capacità di riconoscere nuovi pattern di query.

6.2 SVILUPPI FUTURI

Gli sviluppi futuri di questa tesi sono molteplici:

- Ampliare l'intero sistema in qualcosa di reale progettando e sviluppando un'applicazione mobile (APP) tramite la quale l'ospite si iscrive gratuitamente alla piattaforma "Feel at Home" e compila il proprio profilo, necessario per capire gli interessi dello stesso. In prima battuta, il sistema userà le informazioni del profilo dell'ospite per selezionare un set di "locali" (privati, professionisti, associazioni, piccoli soggetti economici, servizi di pubblica utilità, ...) che offrono servizi e/o suggerimenti concordi con interessi e/o bisogni espressi, anch'essi iscritti e loggati nell'applicazione. Ogni "locale" sarà caratterizzato da una descrizione, una lista di competenze, la forma di contatto che predilige, le lingue parlate e il suo rating (reputazione). L'identità dei soggetti, "locale" ed ospite, sarà validata, attraverso un opportuno processo a stadi, per garantire ai soggetti solo contatti con "identità" certificate. L'ospite potrà così scegliere autonomamente con chi mettersi in contatto aprendo una conversazione con uno o più "locali" scelti, nella quale deve esprimere il suo bisogno e i tempi con cui esso deve essere soddisfatto. Il "locale" potrà accettare o rifiutare la richiesta di contatto che sarà così gestita autonomamente fra i due soggetti coinvolti. Il suggerimento fornito dal "locale" potrà in alcuni casi sfociare in transazioni commerciali tra le parti; tali transazioni non saranno gestite dalla piattaforma, a meno che non facenti parte del catalogo di 'esperienze' gestito dalla piattaforma. Al termine del contatto, entrambi i soggetti saranno invitati a lasciare un feedback riguardante la qualità dell'interazione avvenuta.
- Effettuare uno scraping ulteriormente migliorato prelevando le domande non solo da Yahoo Answers ma anche da altri siti di Query Answering, come ad esempio Quora, in modo tale da raggiungere un numero più elevato di esempi,

andando anche a ampliare le tipologie di intent disponibili, così da rendere più specifica ogni categoria.

- Aumentare il numero di esempi di training andando ad aggiungere automaticamente le nuove query poste dagli utenti finali utilizzatori del sistema. In questo modo, il modello di predizione viene ricalcolato ogni volta che una nuova frase viene inserita in memoria (Active Learning). Questo permetterebbe al modello di funzionare meglio, di restituire una accuratezza migliore e quindi di riuscire a riconoscere nuovi pattern
- Ovvero, attraverso l'intent detection riusciamo a capire quali sono le intenzioni, ovvero se vuole un locale/negozi piuttosto che una persona fisica. Un ulteriore controllo su tale intento è possibile effettuarlo tramite la Topic Recognition che ci permetterebbe di indicare oltre all'intenzione anche il campo di competenza della query. Ad esempio, se la domanda posta è "c'è una palestra per fare allenamento funzionale?", il nostro intent detector ci dirà che l'utente sta richiedendo un locale, una ulteriore rete neurale potrebbe determinare che si tratta di sport, Quindi la query richiede un locale in ambito sportivo. In questo modo è possibile effettuare un filtraggio ulteriore ed è possibile specificare al meglio le caratteristiche della query e soddisfare nel migliore dei modi le esigenze dell'utente.
- Inserire un sistema di ragionamento automatico che basandosi sulle informazioni estrapolate dalla query, riesca a suggerire dei servizi da mostrare all'utente prima dell'interazione con il profilo locale. Ciò permetterebbe a volte di rispondere all'utente prima ancora di mettersi in contatto con una persona fisica. Ad esempio, se l'utente dovesse richiedere una pizzeria in cui mangiare la sera, il sistema dovrebbe proporgli prima di tutto una lista di pizzerie disponibili, concordanti con le sue preferenze, e subito dopo proporgli il contatto di locali con cui parlare e interagire.

RIFERIMENTI BIBLIOGRAFICI:

- [1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781, 2013.
- [2] GloVe: Global Vectors for Word Representation Jeffrey Pennington, Richard Socher, Christopher D. Manning Computer Science Department, Stanford University, Stanford, CA 94305, 2013
- [3] Pratik Jayarao, Aman Srivastava; Intent Detection for code-mix utterances in task oriented dialogue systems, Research And Development Haptik Infotech Pvt Ltd, Mumbai, India; 7 december 2018
- [4] D. Surendran and G. anne Levow, “Dialog act tagging with support vector machines and hidden markov models,” in In Proceedings of
- [5] S. AH, N. Sulaiman, A. Mustapha, and N. Mustapha, “Improving accuracy of intention-based response classification using decision tree,” Information Technology Journal, vol. 8, no. 6, pp. 923–928, 2009.
- [6] Comparative Study of CNN and RNN for Natural Language Processing Wenpeng Yin[†], Katharina Kann[†], Mo Yu[‡] and Hinrich Schutze [‡] † CIS, LMU Munich, Germany [‡] IBM Research, USA
- [7] Conversational Help for Task Completion and Feature Discovery in Personal Assistants; Madan Gopal Jhawar, Vipindeep Vangala, Nishchay Sharma ,Ankur Hayatnagarkar , Mansi Saxena , Swati Valecha, Microsoft India Development Center
- [8] Enriching Word Vectors with Subword Information Piotr Bojanowski* and Edouard Grave* and Armand Joulin and Tomas Mikolov, Facebook AI Research, 2017
- [9] Analysis of Italian Word Embeddings Rocco Tripodi, Stefano Li Pira, 2017
- [10] Convolutional Neural Networks for Sentiment Analysis on Italian Tweets, Giuseppe Attardi, Daniele Sartiano, Chiara Alzetta and Federica Semplici.

- [11] José M. Noguera, Manuel J. Barranco, Rafael J. Segura, Luis Martínez, A mobile 3D-GIS hybrid recommender system for tourism, Elsevier, Information Sciences Volume 215, 15 December 2012
- [12] Cimino A., De Mattei L., Dell'Orletta F. (2018) "Apprendimento multi-task in reti neurali profonde a EVALITA 2018". In Proceedings di EVALITA '18, Evaluation of NLP and Speech Tools per l'italiano, 12-13 dicembre, Torino, Italia.
- [13] Large-Scale Bayesian Logistic Regression for Text Categorization Alexander Genkin, David D Lewis & David Madigan, 2007
- [14] A Review of various k-Nearest Neighbor Query Processing Techniques S. Dhanabal Asst. Professor, Dept. of CSE Dr. S. Chandramathi Professor & Head, Dept. of ECE International Journal of Computer Applications (0975 – 8887) Volume 31– No.7, October 2011
- [15] A C-LSTM Neural Network for Text Classification Chunting Zhou¹, Chonglin Sun², Zhiyuan Liu³, Francis C.M. Lau¹ Department of Computer Science, The University of Hong Kong¹.
- [16] Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning, pages 147–155. Association for Computational Linguistics.
- [17] Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 1030–1038. Association for Computational Linguistics.
- [18] Italy goes to Stanford: a collection of CoreNLP modules for Italian Alessio Palmero Aprosio, Giovanni Moretti.
- [19] <https://blog.cambridgespark.com/tutorial-build-your-own-embedding-and-use-it-in-a-neural-network-e9cde4a81296>
- [20] Deep Bi-Directional LSTM Network for Query Intent Detection, Sreelakshmi Ka, Rafeeqe P Ca, Sreetha Sb, Gayathri E Sb, 8th International Conference on Advances in Computing and Communication (ICACC-2018)
- [21] <https://it.quora.com/Cos%2099%C3%A8-una-rete-neurale-convoluzionale>

[22] Query Intent Detection using Convolutional Neural Networks, Homa B., Amir Asiaee, Reiner Kraft

[23] Understanding LSTM Networks

[24] Bizer, C., Heath, T., Berners-Lee, T., & Sheth, A. P. (2011). Linked data-the story so far. Semantic services, interoperability and web applications: emerging concepts, 205-227.

[25] Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms

[26] Query Intent Determination Using Social Tagging, Colin Shengcai Zhao, 2008

[27] Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms", Thomas Dietterich, 1998

[28] Model evaluation, model selection, and algorithm selection in machine learning, Part IV: Comparing the performance of machine learning models and algorithms using statistical tests and nested cross-validation, Sebastian Raschka, Novembre 2018

RINGRAZIAMENTI

Le prime persone a cui devo dire grazie per questo traguardo sono i miei genitori e la mia famiglia, fonte di sostegno e di coraggio. Senza mia madre e mio padre, non avrei avuto la possibilità di studiare e di scrivere questo elaborato.

Vorrei ringraziare Dott. de Gemmis e il co-relatore di questa tesi di laurea Dott. Polignano, per il supporto che mi ha fornito per la realizzazione di questo progetto di tesi. Grazie a loro ho avuto modo di superare me stesso, acquisendo un prezioso bagaglio di conoscenze che mi aiuteranno nella vita e nel lavoro.

Un grazie speciale ai miei amici, che mi hanno sempre ascoltato e mi hanno sempre sopportato quando in giro ripeteva ad alta voce o quando uscivo con loro per studiare.