Classificazione dei Documenti con CNN-Char, GRNN e HAN

Machine Learning Project

Lorenzo De Luca
Matricola: 7037316
lorenzo.deluca4@stud.unifi.it

Abstract

Nel contesto tecnologico attuale, la Classificazione dei Documenti, è un task fondamentale del Natural Language Processing (NLP), infatti ha il compito di assegnare una classe o una categoria di appartenenza a un documento, rendendolo più facile da gestire e ordinare. In questo articolo trattiamo alcune tecniche di Document Classification, basate sull'apprendimento supervisionato. Replichiamo alcuni dei lavori dell'ultimo decennio utilizzando reti neurali convoluzionali, reti ricorrenti e meccanismi di attenzione. In particolare vengono utilizzati i modelli CNN-Char di Zhang et al. [1], che usano una tecnica tradizionale di *deep learning* per la classificazione del testo, ovvero una Rete Neurale Convoluzionale basata sui caratteri, GRNN di Tang et al. [2], che sviluppa un modello con struttura gerarchica (tramite CNN o LSTM) e basato su un modulo definito Gated Recurrent Neural Network e infine HAN di Yang et al. [3] che tratta un modello con struttura gerarchica con meccanismi di attenzione. I tre modelli vengono valutati su 3 diversi datasets (due di Sentiment Analysis e uno di Topic Classification) e i risultati ottenuti, in linea con quelli descritti nei lavori citati, evidenziano HAN come il modello che raggiunge un'*accuracy* migliore rispetto alle altre tecniche implementate, rispettivamente del 70.1% **e del** 71.5% **su Yelp e Yahoo!Answers.**

1 Introduzione

La classificazione del testo è un compito fondamentale nel *Natural Language Processing*(NLP). Il suo obiettivo consiste nell'assegnare delle etichette al testo in modo tale da identificare l'argomento, classificare il sentimento o individuare eventuale spam. Negli scorsi decenni sono stati utilizzati approcci tradizionali per la classificazione del testo mediante features lessicali sparse, come gli n-grammi, utilizzate per addestrare modelli lineari o basati su kernel, come SVM. Negli ultimi anni, invece, sono state sviluppate tecniche di *deep learning* basate su Reti Neurali Convoluzionali(CNN), Reti Neurali Ricorrenti(RNN) e più recentemente Reti di Attenzione Gerarchica(HAN) ottenendo risultati sempre migliori.
In questo articolo vengono analizzate le più recenti tecniche per la document classification, basandosi sui precedenti lavori di Zhan et al. [1], i quali hanno sviluppato una rete convoluzionale a livello di carattere che possa essere applicata direttamente al testo, senza alcuna conoscenza delle strutture sintattiche o semantiche del linguaggio; di Tang et al.[2] che hanno realizzato un approccio per apprendere la rappresentazione continua del documento, basata sul fatto che un'espressione più lunga(es. una frase o un docu-

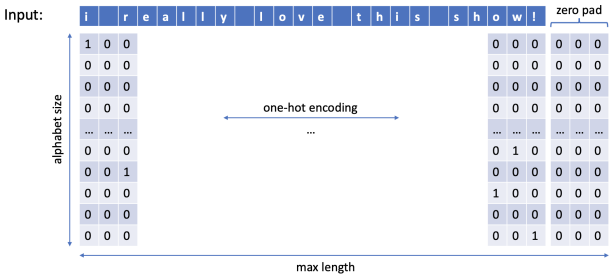

Figure 1: One-hot encoding della frase *"i really love this show!"*

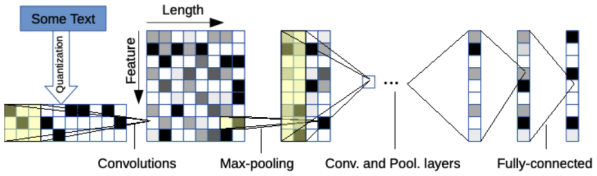

Figure 2: Illustrazione del modello

mento) dipenda dai significati dei suoi costituenti, realizzando un modello che produca una rappresentazione di frasi dalle rappresentazioni delle parole (tramite una CNN o LSTM) e successivamente codificare in modo adattivo la semantica delle frasi e delle loro relazioni intrinseche (tramite una Gated-RNN). Infine i risultati ottenuti sono stati confrontati con il lavoro di Yang et al. [3] i quali si sono posti l'obiettivo di realizzare un modello gerarchico per comprendere l'importanza delle parole e delle frasi, non solo per quello che vogliono dire di per sè, ma anche in relazione al contesto; infatti non tutte le frasi presenti in un documento sono rilevanti allo stesso modo, e scalando di gerarchia, non tutte le parole presenti in un periodo sono rilevanti allo stesso modo. Lo studio sviluppato è stato condotto utilizzando tre datasets di Sentiment Analysis, quali Yelp2013, Yelp2014 e Yahoo!Answer per valutare l'accuratezza dei modelli, ed è reperibile su GitHub[5]

2 Character-level Convolutional Networks

Le Reti Neurali Convoluzionali(CNNs) sono ampiamente utilizzate per sistemi di visione artificiale (*computer vision*), ma hanno fatto la loro comparsa anche in ambito linguistico; infatti proprio come le immagini, i testi possono essere elaborati come la composizione delle loro unità, che non sono pixel, ma caratteri.

2.1 One-hot encoding

L'elemento cardine che ci permette di definire gli input della CNN è la *one-hot encoding*. Grazie alla definizione di un alfabeto di dimensione m che contiene tutti i caratteri che compongono il testo come lettere, cifre, punteggiatura e caratteri speciali e dopo aver fissato una lunghezza massima l_0 di caratteri per ogni frase, possiamo codificare il testo di ciascun periodo, come vediamo in Figura 1. Il testo viene codificato trasformando ogni carattere in un vettore di dimensione pari alla lunghezza dell'alfabeto, con valore 1 in corrispondenza della posizione del carattere nell'alfabeto e 0 nelle rimanenti posizioni; eventuali caratteri maggiori di l_0 vengono ignorati, ma se la frase è composta da un numero di caratteri minore di l_0, allora si aggiunge uno *zero pad*, ovvero tanti vettori di tutti 0 per raggiungere la lunghezza massima.

2.2 Progettazione del modello

La codifica precedente ci permette di quantizzare ogni frase in una matrice binaria, input del modello.
Gli autori in [1] realizzano 2 reti neurali convoluzionali, una grande e una piccola, entrambe a 9 strati (6 convoluzionali + 3 *fully connected*), come è possibile vedere in Figura 2, in cui i moduli chiave sono:

1. **Modulo Convoluzionale temporale**, che calcola una convoluzione 1-D tra una funzione di input discreta $g(x) \in [1, l] \rightarrow R$ e una funzione kernel discreta $f(x) \in [1, k] \rightarrow R$, il cui risultato

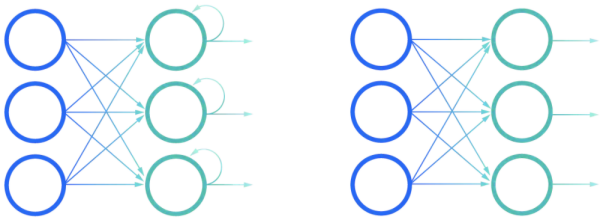

Figure 3: Confronto tra Reti Neurali Ricorrenti (a sinistra) e Reti Neurali Feedforward (a destra)

è dato da

$$h(y) = \sum_{x=1}^{k} f(x)g(yd - x + c) \tag{1}$$

con $h(y) \in [1, \lfloor (l - k + 1)/d \rfloor]$, d passo e $c = k - d + 1$ costante di offset. In particolare, proprio come nelle CNN tradizionali in visione, il modulo è parametrizzato da un insieme di funzioni del kernel $f_{ij}(x)$ che chiamiamo *weights*, su un insieme di inputs $g_i(x)$ e outputs $h_j(y)$ che chiamiamo *features*, di dimensione m(o n) detta *feature size*. Le uscite $h_j(y)$ sono quindi ottenute dalla somma su i delle convoluzioni tra $g_i(x)$ e $f_{ij}(x)$.

2. **Modulo Max-Pooling temporale**, anche questo in versione 1-D, in cui data la funzione di input discreta $g(x) \in [1, l] \to R$, calcola la funzione di max-pooling $h(y) \in [1, \lfloor (l-k+1)/d \rfloor] \to R$ di $g(x)$ come

$$h(y) = \max_{x=1}^{k} g(yd - x + c) \tag{2}$$

I layer *fully connected* finali ci permettono di stabilire, con una determinata probabilità, le varie classi di appartenenza.

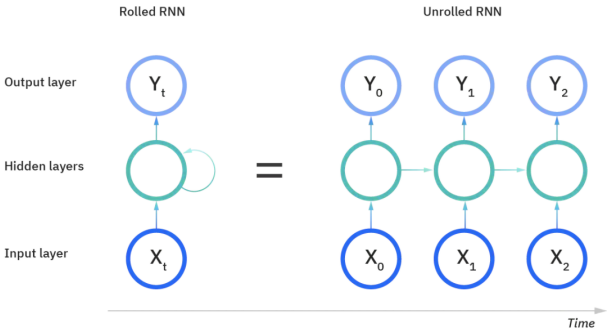

Figure 4: Struttura Rete Neurale Ricorrente *rolled* e *unrolled*

3 Reti Ricorrenti

3.1 Recurrent Neural Networks

Uno nuovo *step* per la NLP, tenta di emulare il comportamento che la mente umana attua per capire un testo, infatti, l'uomo riesce ad interpretare ogni parola del testo in base alla comprensione delle parole precedenti.

Le Reti Neurali tradizionali feedforward non sono in grado di farlo, poiché presumono che input e output siano indipendenti l'uno dall'altro, entrano quindi in gioco le Reti Neurali Ricorrenti (RNNs) in cui l'output dipende dagli elementi precedenti all'interno della sequenza.

Una RNN si distingue da una CNN per la sua memoria, prendendo le informazioni dagli input precedenti per influenzare l'input e l'output correnti; questa proprietà le rende particolarmente adatte a processare delle sequenze temporali, come le parole.

Queste reti possono essere rappresentate mediante dei grafi che contengono dei cicli negli hidden states, oppure visualizzarle nella loro struttura *unrolled*, andando a rappresentare ogni singolo istante di tempo, come si vediamo in Figura 4.

Tuttavia, un problema delle RNN è che durante la backpropagation nel tempo (*BackPropaga-*

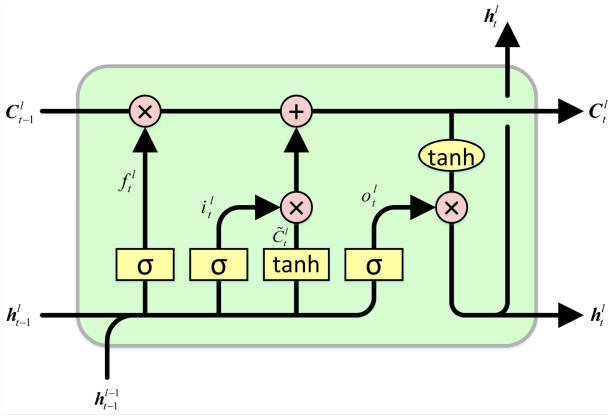

Figure 5: Unità LSTM

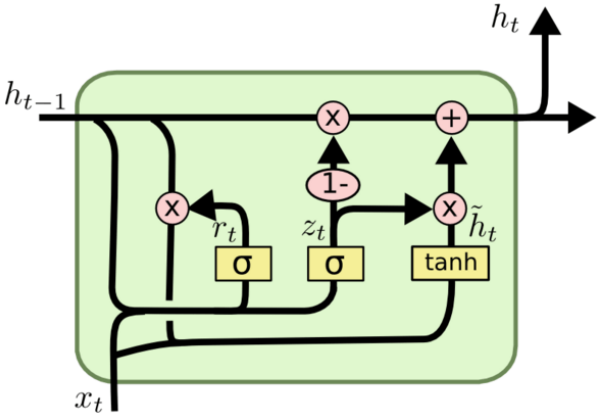

Figure 6: Unità GRU

tion Through Time) per determinare i gradienti, quest'ultimi possono scomparire o esplodere: quando il gradiente è troppo piccolo, continua a ridursi finché non diventa insignificante, e se ciò si verifica, l'algoritmo non apprende più; viceversa l'esplosione del gradiente si verifica quando il gradiente è troppo grande, creando un modello instabile. Questi problemi rendono difficile alla rete imparare le dipendenze a lungo termine tra gli elementi della sequenza [6]. Per evitare questo problema e ottenere un buon addestramento della rete sono stati ideati architetture come LSTM e GRU, usate rispettivamente da Tang et al. in [2] e da Yang et al. in [3].

3.2 Long Short Term Memory

Questa architettura, è stata introdotta da Hochreiter e Schmidhuber [7] come soluzione del *vanishing gradients*, per risolvere il problema delle dipendenze a lungo termine. LSTM, infatti, è in grado di prevedere con precisione lo stato attuale, anche se lo stato precedente che sta influenzando la previsione attuale non è nel passato recente; ricorda quindi sequenze di lunghezza maggiore rispetto a RNN.

Per LSTM, questo è possibile, poiché ha delle celle negli strati nascosti della rete neurale, che hanno tre porte: un *input gate*, un *output gate*, e una *forget gate*. Queste porte controllano il flusso di informazioni necessarie per prevedere l'output nella rete. Più nel dettaglio, la cella presenta uno stato C_t, in cui vengono rimosse o aggiunte informazioni grazie ai *gate*. Ciascun *gate* è attivato da sigmoidi in grado di emettere un output compreso tra 0 e 1, indicando quanto di ciascun componente deve essere lasciato passare [8].

LSTM inizia il suo processo decidendo quali informazioni eliminare dallo stato della cella, ciò viene gestito dalla *forget gate* f_t che calcola la sigmoide tra i suoi pesi e lo stato nascosto precedente h_{t-1} con l'input x_t

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3}$$

dove $f_t = 0/1$ indica rispettivamente di *dimenticare completamente* o *ricordare completamente*.

La seconda decisione, regolata dall'*input gate* i_t stabilisce quali nuove informazioni $\tilde{C}_t$ aggiungere allo stato attuale C_t. Vengono quindi calcolati

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{4}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{5}$$

E successivamente viene aggiornato lo stato C_t:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{6}$$

Infine il nuovo output h_t sarà basato sullo stato della cella C_t filtrato con una tangente iperbolica e regolato dall' *output gate* o_t, calcola quindi

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{7}$$

e genera l'output

$$h_t = o_t * \tanh(C_t) \tag{8}$$

Una rappresentazione del modello è visibile in Figura 5.

3.3 GRU

Questa variante è simile a LSTM, ma invece di utilizzare uno stato della cella per regolare le informazioni, utilizza stati nascosti e, invece di tre porte, ne ha due: una *reset gate* r_t e un *update gate* z_t, che analogamente alle porte di LSTM, controllano la quantità e le informazioni da conservare.

In particolare, ad un certo tempo t, il modulo GRU calcola il nuovo stato nascosto h_t come un'interpolazione lineare tra lo stato precedente h_{t-1} e il nuovo stato corrente $\tilde{h}_t$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{9}$$

Il gate z_t decide quanta informazione passata conservare e quanta nuova informazione aggiungere:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{10}$$

con x_t elemento della sequenza al tempo t.

Lo stato candidato $\tilde{h}_t$ è calcolato in modo simile a una Rete Neurale Ricorrente tradizionale:

$$\tilde{h}_t = \tanh(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h) \tag{11}$$

Il gate r_t controlla quanto lo stato passato contribuisce allo stato candidato, se $r_t = 0$ dimentica lo stato precedente. L'aggiornamento del *reset gate* è calcolato come segue:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{12}$$

Il problema del *vanishing gradient* è diminuito grazie ai gates di controllo, poiché ad ogni istante temporale non viene ricalcolato interamente il prossimo hidden state, ma viene mantenuto lo stato precedente con una piccola modifica rispetto al nuovo input [1].

4 Gated Recurrent Neural Network

Il Gated Recurrent Neural Network(GRNN) è il modello sviluppato in [2], basato sul principio di composizionalità di Frege, il quale afferma che il significato di un'espressione linguistica dipende dal significato delle sue espressioni costituenti. Poiché un documento è costituito da un elenco di frasi, ed ogni frase è costituita da un elenco di parole, il modello viene rappresentato in due fasi:

1. Produzione dei vettori di frase da rappresentazioni di parole (Sentence Level).
2. Uso dei vettori di frase vengono per ottenere la rappresentazione del documento, del quale verrà valutato il sentiment.

In particolare gli autori hanno sviluppato due modelli per la prima fase, attraverso una CNN oppure usando LSTM.

4.1 Word Embedding

In questo caso per definire gli input del modello, utilizziamo il *Word Embedding*. L'importanza del word embedding risiede nel fatto che trasforma le parole in vettori numerici grazie alla definizione di una matrice di embedding, e cattura la semantica e la relazione che intercorre tra le parole.

Quindi, dato un documento, realizziamo un vocabolario V che permetta di assegnare un indice a ciascuna parola univoca; dopodiché codifichiamo l'intero documento andando ad assegnare un indice a ciascuna parola.

Vocabulary	Index coding	Embedding Matrix						
deep	1	.32	.02	.48	...	.21	.56	.15
learning	2	.65	.23	.41	...	.57	.03	.92
is	3	.45	.87	.89	..	.45	.12	.01
very	4	.65	.21	.25	..	.45	.78	.82

Input:	deep	learning	is	very	deep
indices:	1	2	3	4	1
Word Embedding:	.32	.65	.45	.65	.32
	.02	.23	.87	.21	.02
	.48	.41	.89	.25	.48
	...	...	...	...	...
	.21	.57	.45	.45	.21
	.56	.03	.12	.78	.56
	.15	.92	.01	.82	.15

Figure 7: Word Embedding della frase "*deep learning is very deep*"

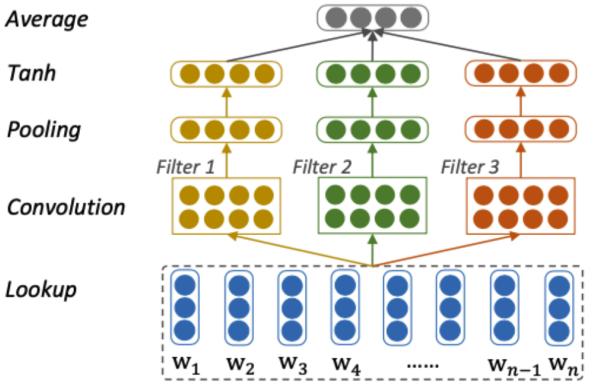

Figure 8: Composizione della frase con CNN

Successivamente, grazie alla realizzazione di una matrice di embedding $L_w \in R^{d \times |V|}$, che può essere addestrata in base al proprio vocabolario oppure usandone una pre-addestrata, vengono associati i vettori numerici a ciascun indice.
Una semplice rappresentazione del processo è raffigurato in Figura 7.

4.2 Progettazione del modello

4.2.1 Sentence Level

Una volta ottenuto il word embedding, questo viene dato in input ad una Rete Neurale Convoluzionale o a LSTM per calcolare la semantica del Sentence Level.

- **CNN:** Questa rete, in Figura 8, comprende tre filtri convoluzionali con *widths* 1,2 e 3 per codificare la semantica di unigrammi, bigrammi e trigrammi in una frase, inoltre ciascun filtro è costituito da un elenco di livelli lineari con parametri condivisi. Quindi presa una frase composta da n parole $\{w_1, w_2, ..., w_n\}$ ciascuna mappata con la propria word embedding $e_i \in R^d$, sia l_c la dimensione (*width*) del filtro convoluzionale, e siano W_c, b_c i parametri condivisi dei livelli lineari nel filtro, l'input del livello lineare è la concatenazione del word embedding di una finestra di dimensione fissata l_c, denotato come $I_c = [e_i; e_{i+1}; ...; e_{i+l_c-1}] \in R^{dl_c}$. L'output del livello lineare è calcolato come

$$O_c = W_c I_c + b_c \tag{13}$$

  Successivamente gli output dei livelli lineari passano attraverso un livello di Average-Pooling, ottenendo un vettore di lunghezza fissa, filtrato con una tangente iperbolica per incorporare la non linearità puntuale e fare la media degli output di più filtri per ottenere la rappresentazione della frase.

- **LSTM:** la struttura di questo livello corrisponde a quella descritta in 3.2

4.2.2 Gated Neural Network

Al termine del layer precedente sono ottenuti i vettori di frase che saranno dati in input ad una Gated Neural Network.
Dati i vettori di frasi di lunghezza variabile come input, la composizione del documento produce un vettore di documento a lunghezza fissa come output, quindi una strategia semplice consiste nell'ignorare l'ordine delle frasi e calcolare la media dei vettori di

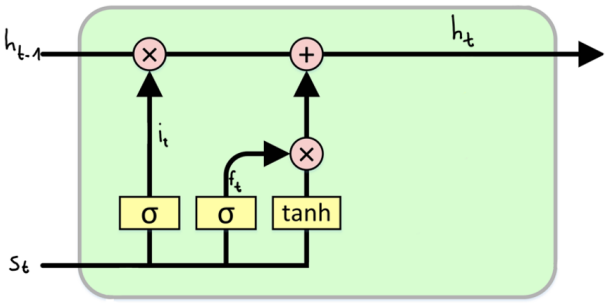

Figure 9: Gated Recurrent Neural Network

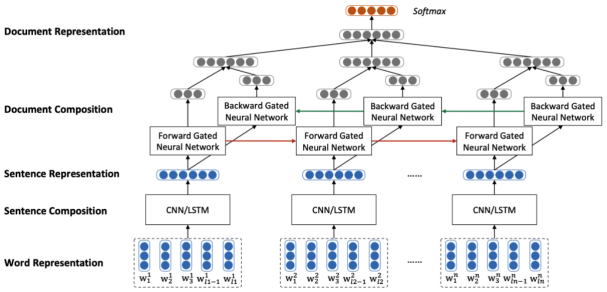

Figure 10: Modello di Rete Neurale a livello di documento per la sentiment classification

frase come vettore di documento. Una RNN standard, come indicato in 3.1 da in output $h(t) = \tanh(W_r[h_{t-1}; s_t] + b_r)$ con s_t vettore di frase corrente e h_{t-1} vettore di output del passaggio precedente. Ma in questo caso, per evitare il problema del *vanishing gradient* viene sviluppata una Gated Recurrent Neural Networks, che funziona in modo sequenziale e codifica in modo adattivo la semantica della frase nelle rappresentazioni del documento.

Nello specifico, la funzione di transizione della GRNN utilizzata, è una variante di LSTM, vedi Figura 9, dove

$$i_t = \sigma(W_i[h_{t-1}, s_t] + b_i) \tag{14}$$
$$f_t = \sigma(W_f[h_{t-1}, s_t] + b_f) \tag{15}$$
$$g_t = \tanh(W_r[h_{t-1}, s_t] + b_r) \tag{16}$$
$$h_t = \tanh(i_t \odot g_t + f_t \odot h_{t-1}) \tag{17}$$

Quindi il modello può essere visto come un LSTM il cui l'*output gate* è sempre attivo. Inoltre per utilizzare le storie precedenti e le evidenze successive allo stesso modo, si sfrutta il GRNN bidirezionale.

Infine, aggiungiamo un livello lineare per trasformare il vettore del documento in un vettore a valori reali la cui lunghezza è il numero di classi C, e un livello Softmax per estrapolare la probabilità di appartenenza ad una certa classe, calcolato come segue:

$$P_i = \frac{\exp(x_i)}{\sum_{i'=1}^{C} \exp(x'_i)} \tag{18}$$

Il modello completo è rappresentato in Figura 10.

5 Hierarchical Attention Network

In questo modello, come trattato da Yang et al. in [3], si riprende la struttura gerarchica, già introdotta da Tang et al. in [2], e si estende con il principio di attenzione.

Questa nuova architettura neurale definita *Hierarchical Attention Network* (HAN), progettata basandosi sul modulo GRU discusso in 3.3, si basa su due intuizioni:

- i documenti hanno una struttura gerarchica (le parole formano frasi, le frasi formano un documento)
- Non tutte le parole in una frase o tutte le frasi in un documento sono ugualmente importanti per la classificazione; inoltre l'importanza delle parole e delle frasi dipende fortemente dal contesto.

5.1 Progettazione del modello

L'architettura complessiva della rete, raffigurata in Figura 11, per includere quanto descritto, è composta da diverse parti: un *Word Encorder*, un *Word Attention*, un *Sentence Encoder* e un *Sentence Attention*.

Supponiamo quindi di avere un documento che contenga L frasi, ciascuna delle quali contenente al mas-

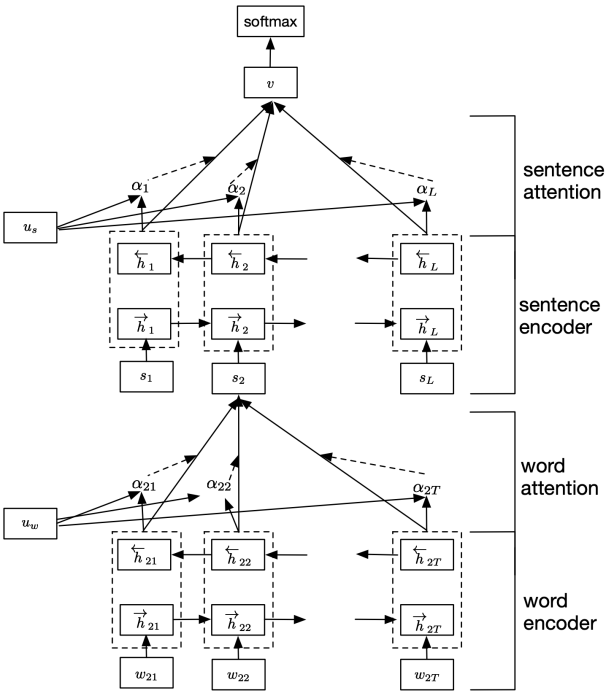

Figure 11: Hierarchical Attention Network

simo T parole, con w_{it} t-esima parola della frase i-esima.
A partire da queste, il primo passo consiste nel definire l'input del modello attraverso il Word Embedding, illustrato in 4.1, ottenendo così il vettore di embedding:

$$x_{it} = W_e w_{it} \tag{19}$$

con W_e matrice di embedding.

5.1.1 Word Level

Word Encoder Si usa una GRU bidirezionale, leggendo quindi le parole in entrambe le direzioni, in modo da ottenere delle annotazioni sul contesto delle parole

$$\overrightarrow{h}_{it} = \overrightarrow{GRU}(x_{it}), t \in [1, T] \tag{20}$$
$$\overleftarrow{h}_{it} = \overleftarrow{GRU}(x_{it}), t \in [T, 1] \tag{21}$$

si ottengono quindi le annotazioni contestuali per una parola w_{it} concatenando i due hidden states, cioè $h_{it} = [\overrightarrow{h}_{it}, \overleftarrow{h}_{it}]$, che vengono date in input al Word Attention.
Word Attention Qui calcoliamo le parole che sono importanti per il significato della frase e si aggrega l'informazione di queste parole formando un *sentence vectors*.

Si costruiscono dei vettori di contesto u_{it} attraverso un Multilayer Perceptron con tangente iperbolica; quindi si misura l'importanza della parola come somiglianza di u_{it} con un *sentence vector* a livello di parola u_w (inizializzata randomicamente), normalizzata attraverso una funzione *softmax*

$$u_{it} = \tanh(W_w h_{it} + b_w) \tag{22}$$
$$\alpha_{it} = \frac{\exp(u_{it}^T u_w)}{\sum_t \exp(u_{it}^T u_w)} \tag{23}$$

poi si calcola il *sentence vector* s_i come somma pesata delle annotazioni

$$s_i = \sum_t \alpha_{it} h_{it} \tag{24}$$

5.1.2 Sentence Level

Sentence Encoder A partire dai *sentence vectors* s_i, ottenuti precedentemente, si ripete il procedimento di encoding con GRU bidirezionale per le frasi

$$\overrightarrow{h}_{i} = \overrightarrow{GRU}(s_i), i \in [1, L] \tag{25}$$
$$\overleftarrow{h}_{i} = \overleftarrow{GRU}(s_i), i \in [L, 1] \tag{26}$$

e si ottiene l'annotazione della frase i, concatenando $\overrightarrow{h}_{i}$ e $\overleftarrow{h}_{i}$, cioè $h_i = [\overrightarrow{h}_{i}, \overleftarrow{h}_{i}]$.
Sentence Attention Per attenzionare le frasi più importanti per classificare il documento si procede in

modo analogo al livello di parola

$$u_i = \tanh(W_s h_i + b_s) \tag{27}$$

$$\alpha_i = \frac{\exp(u_i^T u_s)}{\sum_t \exp(u_i^T u_s)} \tag{28}$$

e si ottiene il vettore v che contiene l'informazione di tutte le frasi del documento:

$$v = \sum_i \alpha_i h_i \tag{29}$$

Infine il vettore v viene utilizzato come features per la classificazione del documento tramite una *softmax*:

$$p = \text{softmax}(W_c v + b_c) \tag{30}$$

6 Esperimenti

Per effettuare un confronto dei modelli descritti nei paragrafi 2, 4 e 5 sono stati condotti alcuni esperimenti durante i quali si è cercato di valutare le prestazioni di ognuno sui diversi datasets.
Gli esperimenti sono stati condotti utilizzando una GPU Nvidia GeForce RTX 2080 Ti 11GB, e l'implementazione del codice è stata effettuata sfruttando Pytorch, un framework di machine learning open source [9].

6.1 Datasets

Valutiamo l'efficacia dei modelli di questo progetto su alcuni datasets di classificazione dei documenti descritti in letteratura, confrontandoli con i risultati ottenuti da Tang et al., Zang et al. e Yang et al.

Questi datasets si dividono in due tipi di attività di classificazione: stima del sentiment, come Yelp2013 e Yelp2014, e classificazione degli argomenti, come Yahoo. Le statistiche dei dataset sono riassunte in Tabella 1.

Yelp è un dataset di recensioni con una valutazione da 1 a 5 stelle. In particolare dopo aver scaricato da [11] tutte le recensioni fino al 2020; sono stati filtrati, tramite uno script Python, le recensioni datate 2013 e 2014.

Yahoo! Answers è un dataset, ottenuto da [12], contenente titolo, domanda e risposta migliore su argomenti generici divisibili in 10 classi: Società e cultura, Scienza e matematica, Salute, Istruzione e consultazione, Computer e Internet, Sport, Affari e finanza, Intrattenimento e musica, Famiglia e relazioni e Politica e Governo.

I dataset utilizzati sono leggermente più piccoli rispetto a quelli di riferimento, nonostante ciò lo split in train set, validation set e test set di ciascun dataset viene eseguito in base alle indicazioni della rispettiva letteratura, in particolare:

- CNN-Char non prevede un validation set, quindi viene eseguito uno split 90/10 per train e test set
- GRNN prevede uno split 80/10/10
- HAN prevede uno split 80/10/10

Dataset	#Class	#Docs
Yelp2013	5	551,715
Yelp2014	5	673,683
Yahoo!Answers	10	1,360,000

Table 1: Statistiche datasets

6.2 Preprocessing

Il preprocessing per CNN-Char, non prevede nessuna elaborazione del testo. E' stato sufficiente definire per la *one-hot encording*, una *max length=1014* un alfabeto di 69 caratteri, che differisce dai 70 caratteri indicati, poiché in [1] vengono inclusi due simboli − uguali: il simbolo meno e il trattino. In GRNN e in HAN vengono suddivisi i documenti in frasi, fino ad un massimo di 20 frasi per documento e viene tokenizzata ciascuna frase in parole (fino a un massimo di 40 parole per HAN) eliminando la punteggiatura

e sostituendo le maiuscole in minuscole. Successivamente viene costruito il vocabolario contenente tutte le parole che compaiono nel dataset in GRNN, mentre in HAN vengono mantenute solo le parole che compaiono almeno 5 volte; eventuali parole non identificate (o nel caso di HAN con un occorrenza minore di 5) vengono sostituite con uno speciale token UNK. Successivamente viene addestrato un modello Word2Vec, grazie ai metodi di Gensim [13], sullo split di train per realizzare l'embedding matrix, con dimensione 200 per ogni word embedding.

Modello	**Yelp 2013**	**Yelp 2014**	**Yahoo!Answ**
	accuracy (*ref_*accuracy)		
CNN-Char(small)	61.0 (-)	64.8 (-)	67.8 (70.1)
CNN-Char(large)	60.0 (-)	62.5 (-)	66.3 (70.4)
GRNN-Conv	61.8 (63.7)	63.2 (65.5)	66.4 (-)
GRNN-LSTM	63.7 (65.1)	65.9 (67.1)	68.6 (-)
HAN	**68.1** (68.2)	**70.1** (70.5)	**71.5** (75.8)

Table 2: Risultati sul test sets. *ref_accuracy* indica i risultati dei lavori in [1], [2] e [3]

6.3 Training

In CNN-Char sono state addestrate due reti, una piccola e una grande, le quali differivano dal numero di canali in ingresso negli strati convoluzionali (256 per quella piccola e 1024 per quella grande) e dalla dimensione di output dei layer fully connected (1024 per quella piccola e 2048 per quella grande). L'ottimizzatore scelto è stato Stochastic Gradient Descent, con learning rate 0.03 e momentum 0.9.

Anche in GRNN, vengono inizializzate due reti, come illustrato precedentemente, per calcolare la semantica del Sentence Level: la prima svolge questa fase con una rete neurale convoluzionale, mentre la seconda usa LSTM. Anche in questo caso l'ottimizzatore scelto è stato Stochastic Gradient Descent.

7 Analisi dei Risultati

Al termine degli esperimenti di classificazione del sentiment a livello di documento i risultati ottenuti sono quelli riportati in Tabella 2. La tabella raccoglie i valori di accuratezza raggiunti dai vari modelli implementati, facendo un confronto anche con i valori di accuratezza raggiunti dai lavori originali di riferimento, ottenendo dei risultati pressoché simili.

CNN-Char, utilizzando metodi meno all'avanguardia, raggiunge risultati peggiori rispetto ai modelli successivi, nonostante ciò, come osservato anche in Zhang et al., per dataset più grandi e con testi meno curati, quindi con alcuni errori ortografici o slang, tipo Yahoo!Answers, ha un funzionamento migliore, che si avvicina maggiormente ad HAN (che ha l'accuratezza migliore). Per quanto riguarda il confronto tra la rete small e quella large, non si notano sostanziali differenze, con prestazioni leggermente migliori per la rete small.

HAN è la rete più articolata, infatti unisce oltre alle reti ricorrenti i meccanismi di attenzione, e raggiunge come previsto i migliori risultati su tutti e 3 i datasets. In particolare migliora le CNN circa del 6-7% e le GRNN del 4-5% su Yelp, tuttavia si nota un notevole calo di prestazione in Yahoo!Answers, probabilmente dovuto al fatto che molte parole vengono scartate e identificate come UNK (sconosciute) a causa della struttura del dataset che tende ad essere formato da errori grammaticali. Nonostante ciò, probabilmente grazie al meccanismo di attenzione, riesce a mantenere il miglior risultato anche su questo dataset.

Infine il modello GRNN, in cui viene implementata solo una struttura gerarchica tramite le reti ricorrenti, riesce ad ottenere risultati migliori di CNN-Char di circa il 2%, nei dataset di sentiment analysis, ma ottiene prestazioni più scadenti su Yahoo, probabilmente dovuto, come per HAN, alla composizione di un dataset con una grammatica non perfettamente corretta. Anche in questo caso le due reti realizzate, Convoluzionale e LSTM, non discostano eccessivamente nei risultati, ma si nota comunque un accuratezza migliore per la rete realizzata con LSTM.

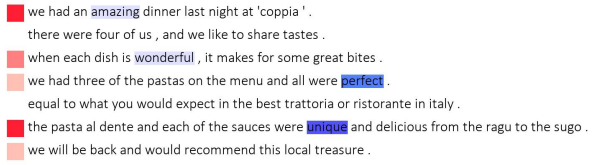

Figure 12: Recensione a 5 stelle: la prima e la quarta frase danno un significato più forte e le parole "amazing" e "unique" contribuiscono maggiormente a definire il sentimento delle due frasi (Previsione: 5).

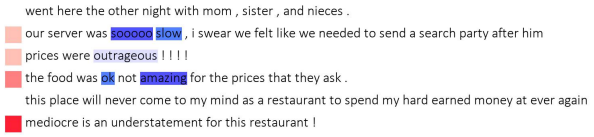

Figure 13: Recensione a 1 stella: l'ultima frase dà il significato più forte, ma le parole "sooo slow" e "amazing" in questo contesto aiutano a definire il sentimento negativo del documento (Previsione: 1).

7.1 Visualizzazione dell'attenzione in HAN

Per mostrare che HAN è in grado di selezionare frasi e parole informative in un documento, visualizziamo i livelli di attenzione gerarchica nelle Figure 12 e 13, in cui sono rappresentati dei documenti dal dataset Yelp 2013. Le Figure rappresentano il documento scomposto in frasi, in cui ogni riga rappresenta la singola frase. I colori, invece, indicano l'importanza di ciascun elemento con diverse sfumature di colore, in particolare il rosso indica il peso della frase e il blu indica il peso della parola.
Dai risultati ottenuti si può notare come il modello riconosca piuttosto bene le parole più significative. Nello specifico le parole classificate "*importanti*" assumono un diverso peso, in negativo o in positivo, anche a seconda del contesto: negli esemi riportati mostriamo come la parola "*amazing*" assuma un importanza positiva in una recensione a 5 stelle, ma negativa se anticipata da "not". Inoltre come ci aspettiamo, parole o frasi non interessanti all'identificazione positiva o negativa del documento non vengono evidenziate in nessun modo, e possono quindi essere identificate come elementi neutrali del documento.

8 Conclusione

In questo lavoro è stato affrontato il problema della classificazione dei documenti in modo supervisionato, prendendo come riferimento dataset di Sentiment Analysis e di classificazione degli argomenti, partendo da modelli che si basassero esclusivamente sulle Reti Neurali Convoluzionali, proseguendo con modelli che esplorassero solo la struttura gerarchica attraverso Reti Neurali Ricorrenti fino ad arrivare a modelli più complessi come HAN, che combinano il meccanismo dell'attenzione con la struttura gerarchica.

I risultati sembrano mostrare HAN come netto favorito rispetto ai metodi precedenti, come GRNN e soprattutto CNN-Char; infatti la costruzione di un vettore documento costruito progressivamente da vettori di frase e vettori di parole importanti porta ad avere i migliori risultati, indipendentemente dal datasets e dal tipo di classificazione (sentimento o argomento).

Tuttavia anche esplorare solo la struttura gerarchica, come GRNN (soprattutto LSTM-GRNN), può migliorare i risultati rispetto a CNN-char, che comunque ha un rendimento molto variabile in base alla dimensione e la cura del testo nel dataset.

References

[1] Xiang Zhang, Junbo Zhao and Yann LeCun. 2015. Character-level convolutional networks for text classification. In Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, pages 649–657.

[2] Duyu Tang, Bing Qin and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of the 2015 Conference on

Empirical Methods in Natural Language Processing, pages 1422–1432.

[3] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1480–1489.

[4] Duyu Tang, Bing Qin and Ting Liu. 2015. Learning Semantic Representations of Users and Products for Document Level Sentiment Classification.

[5] Repository GitHub del progetto `https://github.com/loredeluca/DocClassification-with-CNN-Char-GRNN-and-HAN`.

[6] IBM Cloud Education. 2020. Recurrent Neural Networks.

[7] S.Hochreiter J.Schmidhuber. 1997. Long Short-Term Memory. Neural Computation.

[8] C. Olah. 2015. Understanding LSTM Networks. Colah's Blog.

[9] Pytorch Freamework. `https://pytorch.org`.

[10] J. Brownlee. 2017. What Are Word Embeddings for Text? Machine Learning Mastery.

[11] Yelp Datasets. `https://www.yelp.com/dataset`.

[12] Yahoo Datasets. `https://www.kaggle.com/soumikrakshit/yahoo-answers-dataset`.

[13] Gensim, topic modelling for humans. `https://radimrehurek.com/gensim/intro.html`.

[14] Repository Tutorial Text Classification. `https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Text-Classification`.