Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Progetto Finale

Deep Matching di inserzioni su piattaforme di E-Commerce

Autori:
Gianluca Scarpellini - 807541- g.scarpellini1@campus.unimib.it
Federico Belotti - 808708 - f.belotti8@campus.unimib.it

24 gennaio 2020

Sommario

La diffusione su scala mondiale dei servizi di E-Commerce ha generato un mercato dal valore stimato per il 2020 di circa 3.9 trilioni di dollari e per il 2021 di 4.5 trilioni di dollari. Aziende come Amazon, AliExpress, Ebay producono annualmente terabye di dati e hanno necessità di convertire i dati in informazioni di valore. Uno dei problemi che una grande mole di dati tra loro eterogenei pone riguarda il matching tra entità (detto **duplicate detection** o **link discovery**). Nel presente lavoro analizziamo ed estendiamo tecniche di NLP e modelli di Deep Learning allo scopo di risolvere il matching di inserzioni impiegando solo i titoli delle stesse.

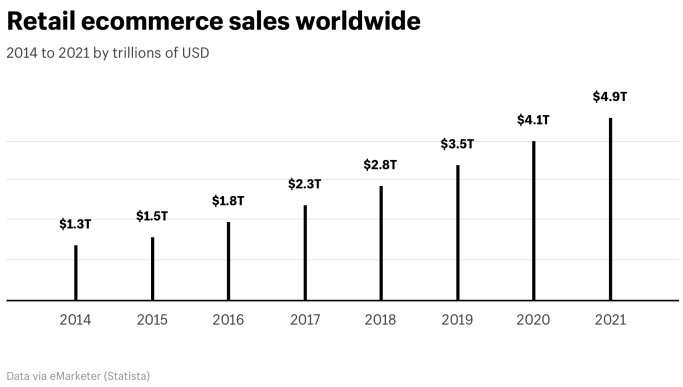

Figura 1
Mercato dell'E-Commerce *(fonte: Shopify)*

1 Introduzione

La ricerca nel settore di **link discovery** è oggi incentrata in metodi di matching che fanno uso di rappresentazioni vettoriali (o **embeddings**) e dei recenti sviluppi del Machine Learning, in particolare di modelli di Deep Learning quali Recursive Neural Network e Convolutional Neural Netowrk. Il dataset da noi scelto, il **The WDC Training Dataset and Gold Standard for Large-Scale Product Matching** [1], mette a disposizione un corpus

di titoli di inserzioni ottenuto dallo *scraping* e dall'annotazione di milioni di offerte sul web. In sezione 2 presentiamo il dataset impiegato per addestrare e testare i modelli; lo stesso dataset è stato impiegato anche per la generazione degli embeddings. In sezione 3 presentiamo le principali tecniche di NLP, i modelli di embeddings e il modello di Deep Learning impiegato per il task di regressione logistica. In sezione 4 riportiamo i risultati degli esperimenti effettuati; una discussione dettagliata degli stessi è presentata in sezione 5, in cui valutiamo il nostro approccio confrontandoci con lo stato dell'arte. In sezione 6 sono infine riportate le nostre considerazioni circa l'approccio scelto e possibili sviluppi futuri dello stesso.

2 Dataset

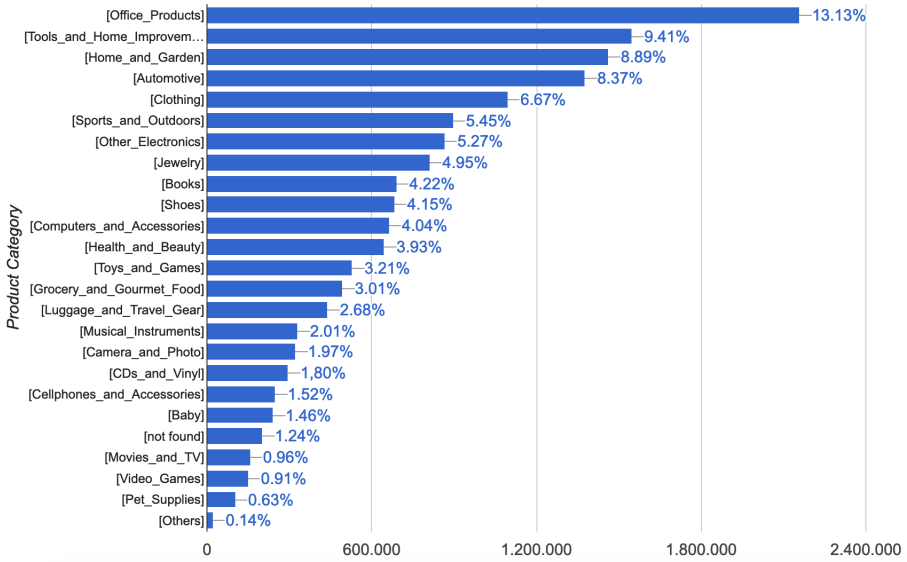

Figura 2
Entità presenti nel dataset

Abbiamo deciso di impiegare **WDC Product Data Corpus e il Gold Standard for Large-scale Product Matching** [1] quale dataset per addestrare e validare il nostro approccio. Il corpus consiste di 26 milioni di inserzioni provenienti da 79 mila siti differenti. Le offerte sono raggruppate in 16 milioni di cluster; ciascun cluster si rifersce allo stesso prodotto usando

degli identificatori univoci, come i GTIN oppure MPN. Il **gold standard** consiste di 4.400 coppie di offerte di cui è indicato il **matching** (ossia, un valore binario **True** oppure **False** indicante l'uguaglianza delle due offerte in termini di prodotto). In figura 2 sono riportate le categorie di offerte e il relativo istogramma di frequenza. In tabella 1 sono riportate le informazioni a nostra disposizione per ciascun'entità. Il task che ci proponiamo di risolvere prevede che, data una coppia di entità così strutturate, il modello sia in grado di predirne il **matching**: ciascun elemento di training è quindi una tripla composta da due entità, ossia due inserzioni, e un **target** binario che indichi se queste si riferiscono allo stesso prodotto. WDC propone un dataset costituto da *positive e negative samples* suddiviso per le categorie principali di prodotti (si faccia riferimento alla figura 2). Abbiamo deciso di adottare il comparto **computers** quale dataset di addestramento[1]. Esso, pur rappresentando il 4.4% del totale, è costituito da oltre 60.000 entità. Abbiamo infine impiegato il **gold standard** fornito da WDC per validare la bontà del nostro approccio.

Tabella 1
Riassunto attributi presenti per ogni inserzione

Attributi	Contenuto
id	Identificatore univoco di un'inserzione
category	Una delle 25 categorie di prodotti
title	Titolo dell'inserzione
description	Descrizione del prodotto data dall'inserzionista
brand	Brand del prodotto
price	Prezzo del prodotto in inserzione
specTableContent	Dettagli e informazioni aggiuntive quali url dell'offerta, etc.

In particolare, abbiamo impiegato due versioni di dataset tra quelle proposte. Le due versioni, denominate **medium** e **large**, si differenziano per il numero di esempi positivi e negativi presenti. Una panoramica di tali differenze è presentata in figura 2. La composizione multi-scala è ottenuta, riprendendo da [1], mantenendo un rapporto di 1:3 tra **positive** e **negative samples**.

[1]Si faccia riferimento alla pagina ufficiale `http://webdatacommons.org/largescaleproductcorpus/v2/index.html`

Tabella 2
Composizione del dataset

	Dataset		
	positive	negative	combined
medium	1,762	6,332	8,094
large	6,146	27,213	33,359
gold standard	800	300	1,100

3 Approccio

La pipeline che abbiamo impiegato fa uso delle **rappresentazioni vettoriali** di ogni vocabolo (o token) presente nella coppia di titoli. I due titoli, rappresentati separatamente da una lista di vettori, uno per ogni token presente in essi, vengono elaborati da due reti **LSTM**, una per titolo, per ottenere una rappresentazione che che tenga in considerazione la sequenzialità del testo in input; successivamente, si compone una **matrice di similarità** che viene elaborata spazialmente da una **Convolutional Neural Network** (CNN). In coda alla rete è posto un **Multi-layer Perceptron** (MLP) al fine di rispondere al task di regressione binaria.

Un approccio di questo tipo si è dimostrato particolarmente efficace in task di matching tra prodotti, come riportato in [2]. Nelle sezioni 3.1 e 3.2 è esposto il nostro approccio per la costruzione di vettori significativi per il task in oggetto a partire dal corpus della coppia di titoli. In sezione 3.3 sono presentate 2 tipologie di matrici di similarità da noi implementate. Infine, in sezione 3.4 si definisce il modello di regressione logistica e si riportano le specifiche del nostro approccio di ricerca degli iperparametri. Crediamo nell'open-source quale opportunità di crescità per la community di Machine Learning; abbiamo pertanto deciso di distribuire l'implementazione della pipeline con licenza Apache License 2.0[2].

3.1 Preprocessing e Tokenizzazione

Il primo step del nostro approccio prevede una fase di **preprocessing** del corpus di input. Abbiamo sviluppato una pipeline di preprocessing che, dato

[2]Disponibile a `https://github.com/belerico/insertion-matcher`

in input un titolo testuale, restituisca la lista di **token**[3] di cui esso è composto. Al titolo in input vengono applicati dei filtri al fine di rimuovere segni di punteggiatura e stopwords. I token dei due titoli vengono successivamente mappati tramite un **word index**, che associa a ogni vocabolo un indice intero univoco. Un parametro **max_len** definisce infine la dimensione che ciascuna sequenza deve assumere: ci occupiamo pertanto di riportare ciascun titolo a una dimensione standard, aggiungendo indici nulli o troncando la coda del corpus.
Ad esempio:

```
"495906 b21 hp x5560 2 80ghz ml350 g6 , null new
    wholesale price"
                              ↓
['495906', 'b21', 'hp', 'x5560', '2', '80ghz', 'ml350
    ', 'g6', 'new', 'wholesale', 'price']
                              ↓
[3240, 891, 2, 782, 1374, 400, 752, 3254, 45, 321,
    26, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

dove il numero 1 rappresenta l'intero di padding.

3.2 Embeddings e RNN

Abbiamo deciso di impiegare diverse versioni di embeddings al fine di valutare la bontà del nostro algoritmo. In particolare, abbiamo deciso di confrontare entrambi **word2vec** e **fasttext** [3, 4, 5]. Sia fasttext che wor2vec sono due modelli predittivi che apprendono rappresentazioni vettoriali a partire da un corpus scritto in linguaggio naturale, sfruttando la cosìdetta **distributional hypothesis**: parole simili tendono ad apparire in contesti simili. Per entrambi i modelli esistono due differenti metodologie di definizione del problema; data una finestra di contesto di dimensione fissa, che viene fatta scorrere su tutto il corpus di input, si avrà:

- **CBOW**: il modello viene addestrato per predire una parola target, dato il contesto che la circonda

[3] `https://www.nltk.org/`

- **SkipGram**: il modello viene addestrato, al contrario di CBOW, a predire le parole che appaiono nel contesto di una parola target

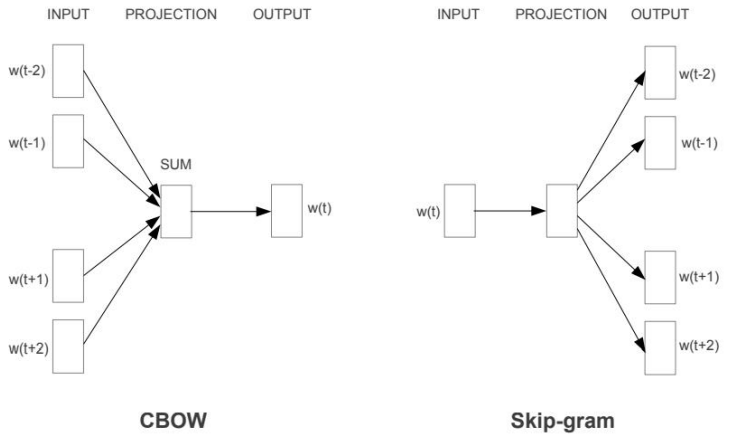

Figura 3
CBOW e SkipGram

Per rendere efficiente l'apprendimento, il problema viene trasformato in un problema di classificazione in cui il modello deve predire correttamente se una coppia di parole appartengono allo stesso contesto o meno, da cui la necessità di introdurre esempi negativi in modo randomico (**Negative Sampling**) per rendere più robusto il modello [3, 4]. Fasttext, in aggiunta, arricchisce le rappresentazioni vettoriali apprese tenendo in considerazione anche le "subword", ovvero tutti gli n-gram di caratteri, con n variabile, di cui è composta una parola, permettendo così di rappresentare parole non presenti nel vocabolario sommando le rappresentazioni degli n-gram che la compongono [5].

Entrambi i modelli sono dunque stati addestrati per 300 epoche, con la metodologia SkipGram, una finestra di contesto pari a 9 e con il sampling di 10 esempi negativi per ogni esempio positivo; abbiamo addestrato entrambi i modelli per apprendere rappresentazioni vettoriali di **100**, **150** e **200** dimensioni. Non abbiamo inoltre permesso al modello di modificare le rappresentazioni apprese, mantenendole dunque fisse durante la fase di apprendimento.

Ciascuna rappresentazione vettoriale è input di una LSTM [6], un modello molto utilizzato per la sua capacità nel rappresentare sequenze temporali,

siano esse un testo scritto o una sequenza audio, per estrarre un vettore che tenga in considerazione la sequenzialità intrinseca dell'input. Abbiamo deciso di impiegare questo modello di RNN basandoci sullo stato dell'arte [2].

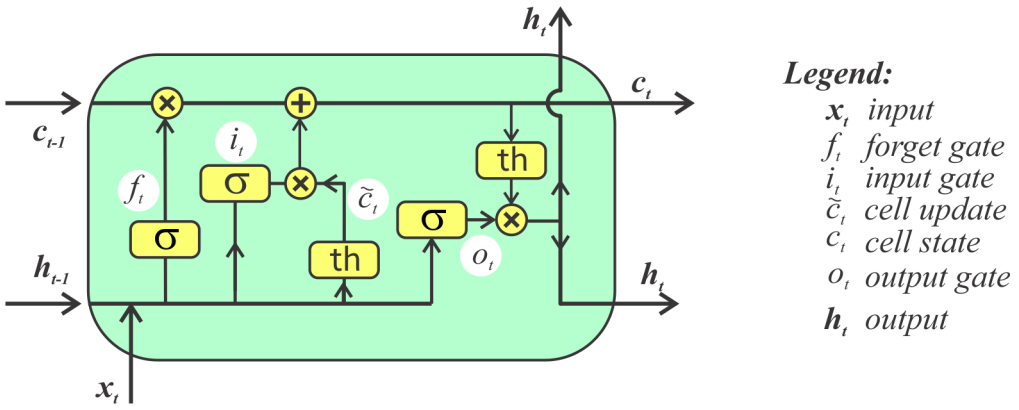

Figura 4
LSTM

3.3 Matrici di similarità

In sezioni 3.1 e 3.2 si è definito il processing del corpus dalla forma testuale a una forma vettoriale adatta al task di **similarity matching**. A tale scopo, abbiamo deciso di adottare un approccio simile a quello presentato in [2]. I due titoli, rappresentati dalle sequenze di vettori di lunghezza massima max_len (una per token o vocabolo), dopo essere stati processati separatamente da due reti LSTM distinte, vengono combinati in una matrice di similarità di dimensione max_len × max_len, dove la poszione ij rappresenta la similarità tra la rappresentazione vettoriale del token i-esimo e quella del token j-esimo, del primo e secondo titolo rispettivamente. Abbiamo implementato 2 tipi di funzioni di similarità tra vettori, le quali sono già impiegate in applicazioni allo stato dell'arte:

- **Dot similarity**: per ogni coppia di vocaboli, si valuta la similarità delle rappresentazioni vettoriali come il prodotto scalare tra esse
- **Cosine similarity**: per ogni coppia di vocaboli, si valuta la similairità delle rappresentazioni vettoriali come il coseno dell'angolo tra esse

3.4 CNN e MLP

Il passaggio finale del nostro modello impiega una Convolututional Neural Network (CNN) composta da un unico layer di convoluzione, seguito da un layer di batch normalization e uno di max pooling, per elaborare spazialmente la matrice di similarità, così da apprendere le interazioni tra i token dei due titoli, e porre dunque attenzione a quei vocaboli che possono meglio rappresentare l'input; l'output della CNN viene riportato a un tensore *piatto* in una dimensione; quest'ultimo è infine impiegato come input di un MLP (multi-layer perceptron), composto da due layer, seguiti entrambi da due layer di dropout.
In testa al modello è posto un neurone dotato di attivazione *softmax*: l'obiezione comune consiste nell'affermare che, essendo un problema di classificazione binaria, un'attivazione di tipo *sigmoid* può essere più che sufficiente per ottenere uno score di confidenza tra 0 e 1, da trasformare poi in 0 (non-match) e 1 (match) mediante l'utilizzo di una soglia. Il vantaggio di un sistema di classificazione simile consiste sia nel poter apprendere quale sia la soglia migliore sia nella possibilità di controllare Precision e Recall "più o meno" a piacimento. Per evitare di dover apprendere tale soglia, o di utilizzarne una fissa, abbiamo permesso al modello di esprimere il livello di confidenza per entrambe le classi, e in fase di predizione di scegliere quella più probabile, da cui l'utilizzo della softmax come attivazione dell'ultimo layer.

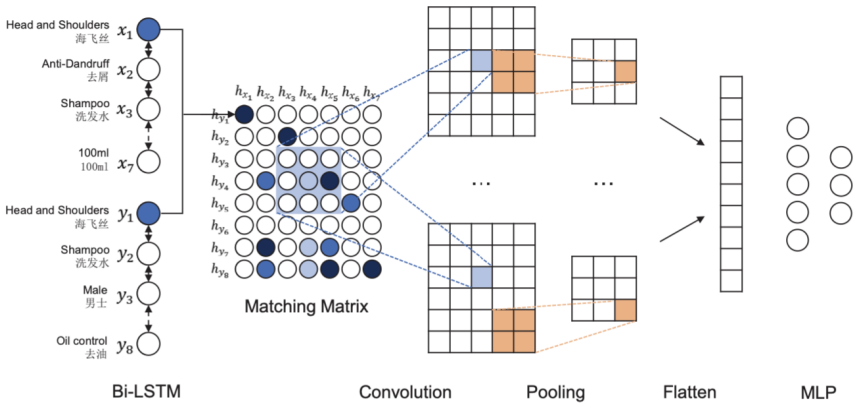

Figura 5
Modello completo

3.5 Ottimizzazione degli iperparametri

Il modello, pur essendo architetturalmente non troppo complesso, lo è nei possibili iperparametri che lo caratterizzano. Abbiamo dunque deciso di ottimizzarne alcuni, quelli secondo noi più rilevanti:

Tabella 3
Iperparametri e loro dominio

Iperparametri	Tipo	Dominio
learning rate	$\mathbb{R}$	[1e-6, 1e-3]
rnn units	$\mathbb{N}$	$[50, 250]$
convs banks	$\mathbb{N}$	$[4, 64]$
convs kernel size	$\mathbb{N}$	$[2, 3]$
dense units 1	$\mathbb{N}$	$[16, 128]$
dense units 2	$\mathbb{N}$	$[16, 128]$
similarity type	$\mathbb{N}$	[dot, cosine]

Abbiamo dunque deciso di procedere come segue:

1. Abbiamo diviso il dataset *medium* in due: *medium train* e *medium val*, in un rapporto specificato dal WDC (si faccia riferimento a sezione 2 per la discussione sul dataset);

2. Per ogni tipologia di embedding e per ogni dimensione, abbiamo ottimizzato gli iperparametri in tabella 3 utilizzando un **modello surrogato di tipo gaussiano** con funzione d'acquisizione di tipo **Expected Improvement**. Le valutazioni compiute sono 10, ognuna da 10 epoche di addestramento. I modelli sono stati addestrati utilizzando i dataset *medium train* e *medium val*

3. Appresi gli iperparametri migliori, viene addestrato un modello sul dataset *medium* per 30 epoche e vengono valutate le performance sul **gold standard**;

4. Gli stessi iperparametri vengono utilizzati per addestrare un modello sul dataset *large* per 30 epoche, valutandone le performance sul **gold standard**;

Per ogni esperimento, abbiamo impiegato come ottimizzatore **Adam**, il dropout nell'MLP è stato fissato a 0.3 e un **batch size** da 32 esempi, affidandoci sulla bontà di queste scelte e sullo stato dell'arte.

4 Esperimenti

In tabella 5 sono riportati i risultati ottenuti sul *gold standard* addestrando il nostro modello con i migliori iperparametri sul dataset *medium*. In particolare, per entrambi i modelli di embeddings impiegati sono riportati in tabella 4 gli iperparametri migliori ottenuti mediante il modello surrogato di tipo gaussaino, suddivisi per dimensione dell'embedding. In tabella 6 sono invece riportati i risultati ottenuti, come al punto precedente, addestrando il modello con i migliori iperparametri sul dataset *large*.

Tabella 4
Iperparametri ottenuti tramite ricerca **gaussiana**

	100d	150d	200d
Word2Vec			
learning rate	1e-3	2e-3	1e-3
rnn units	200	154	200
convs banks	32	61	32
convs kernel size	3	2	2
denses units 1	32	99	32
denses units 2	16	44	32
similarity type	dot	cos	dot
fasttext			
learning rate	5e-4	7e-4	6e-4
rnn units	244	136	104
convs banks	7	61	62
convs kernel size	2	2	2
denses units 1	84	74	94
denses units 2	84	74	94
similarity type	dot	dot	dot

Tabella 5
Risultati degli esperimenti su dataset *Medium* ottenuti con i parametri migliori

	P %	R %	F1 %
Word2Vec			
100	75	76	75
150	73	72	72
200	78	76	77
fasttext			
100	77	76	76
150	76	76	76
200	72	72	72

Tabella 6
Risultati degli esperimenti su dataset *Large* ottenuti con i parametri migliori

	P %	R %	F1 %
Word2Vec			
100	85	84	84
150	83	83	83
200	85	85	85
fasttext			
100	84	84	84
150	86	85	85
200	84	84	84

5 Analisi dei risultati

Nelle tabelle 7 e 8 vengono messi a confronto i risultati del nostro approccio con lo stato dell'arte in ambito *matching* sui dataset *medium* e *large*. La ricerca effettuata tramite **processi gaussiani** (si veda sezione 3.5) ci ha permesso di individuare i migliori parametri utilizzando solo un sottoinsieme del dataset a disposizione. Nelle tabelle 5 e 6 sono riportati i risultati ottenuti su ogni embedding impiegando i parametri indicati in tabella 4. In particolare, l'addestramento sul dataset medium ci ha portato a prediligere embedding

di dimensione **100** e **200**. Per quest'ultimi infatti le misure di riferimento **precision**, **recall** e**fscore** sono risultate più alte e al di sopra dello stato dell'arte, in particolare con i modelli addestrati tramite *word2vec*.

L'addestramento sul dataset *large* ha portato a risultati più soddisfacenti su embedding di dimensione **150** e **200**. Il risultato migliore si è ottenuto impiegando un'embedding di dimensione **150** estratto tramite *fasttext*. Dal confronto con la tabella 8 mettiamo in luce la bontà del nostro approccio contro alcuni modelli di *product matching*: il nostro approccio si confronta bene con i modelli di *machine learning* più evoluti, superando per tutte le misure i modelli *magellan* e *co-occ*, pur mantenendosi al di sotto delle performance di *deepmatcher*[4].

La bontà di questo approccio risiede nella semplicità architetturale, che è stata resa poco più "leggera" di quella utilizzata in [2], e nella giusta combinazione di elementi singoli di cui è composto: le LSTM giocano un ruolo fondamentale per l'interpretazione di input sequenziali, mentre il layer convoluzionale pone "attenzione" ai quei token che sono più indicativi della similarità di due prodotti (ad esempio focalizzandosi su quei token che rappresentano gli identificativi dei prodotti), infine il MLP prende la decisione più probabilmente adatta.

6 Conclusione

Il settore dell'E-commerce è oggi un mercato in esplosione. Le grandi piattaforme di domanda/offerta avranno in futuro sempre più necessità di associare inserzioni sulla base del prodotto che esse espongono. Questo tipo di task va incontro alla necessità di costruire *graph knowledge* dei prodotti presenti sulla piattaforma stessa e diventano essenziali qualora si volesse interpretare le abitudini dei consumatori. L'approccio esposto nel presente lavoro mette in pratica i più recenti sviluppi sullo stato dell'arte nell'ambito del **product matching**. Nelle tabelle 7 e 8 abbiamo confrontato il migliore tra i nostri modelli addestrati sul dataset *medium* e *large* e validati su un **gold standard** rappresentativo. Crediamo che quest'approccio e la pipeline da noi implementata potrebbe essere impiegata a livello di produzione per l'addestramento di modelli orientati a massimizzare le performance su **precision, recall, fscore**.

[4] `http://webdatacommons.org/largescaleproductcorpus/v2/index.html#toc5.2`

Tra i possibili sviluppi futuri, ipotizziamo la possibilità di estendere li modello in profondità, sfruttando lo stato dell'arte in ambito CNN, e l'estensione ai moderni approcci di Natural Language Processing. Basandoci inoltre sulla nostra esperienza umana, un altro possibile sviluppo futuro consiste nell'invertire l'ordine del processing: applicare prima le convoluzioni alle rappresentazioni vettoriali del testo, così come noi esseri umani prima guardiamo il testo che abbiamo di fronte, processare poi il risultato con le reti LSTM e prendere infine una decisione.

Riferimenti bibliografici

[1] A. Primpeli, R. Peeters, and C. Bizer, "The wdc training dataset and gold standard for large-scale product matching," in *Companion Proceedings of The 2019 World Wide Web Conference*, ser. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 381–386. [Online]. Available: https://doi.org/10.1145/3308560.3316609

[2] J. Li, Z. Dou, Y. Zhu, X. Zuo, and J.-R. Wen, "Deep cross-platform product matching in e-commerce," *Information Retrieval Journal*, Aug 2019. [Online]. Available: http://dx.doi.org/10.1007/s10791-019-09360-1

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *CoRR*, vol. abs/1607.04606, 2016. [Online]. Available: http://arxiv.org/abs/1607.04606

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

Tabella 7
Confronto dei risultati con lo stato dell'arte per il dataset *medium*[1]. Sono riportati, per ogni metodo, le *features* impiegate

	P %	R %	F1 %
TFIDF-cosine *title+desc+brand*	42	89	57
Magellan *title+desc+brand+specTable*	50	87	64
Co-Occ *title+desc+brand+specTable*	68	82	74
Deepmatcher *title*	71	85	77
Our approach *title*	78	76	77

Tabella 8
Confronto dei risultati con lo stato dell'arte per il dataset *large*[1]. Sono riportati, per ogni metodo, le *features* impiegate

	P %	R %	F1 %
TFIDF-cosine *title+desc+brand*	42	89	57
Magellan *title+desc+brand+specTable*	69	69	69
Co-Occ *title+desc+brand+specTable*	83	79	81
Deepmatcher *title*	86	92	89
Our approach *title*	86	85	85