Question Classification and On-Line Alghoritms: Winnow Classifier

Valerio Barbagallo & Fabio Grucci

8 settembre 2008

Sommario

L'articolo tratta del problema del Question Answering (QA) in particolare della Question Classification (QC).

Nel dettaglio vengono trattati i problemi che sorgono quando si vogliono realizzare algoritmi per la QC: analisi del dominio di interesse, scelta di una tipologia di algoritmo (on-line/off-line), scelta dell'algoritmo, scelta delle feature da applicare, implementazione e confronto con altri classificatori noti.

In modo particolare l'interesse è quello di testare l'efficacia di un algoritmo on-line (Winnow) nel dominio della Question Classification, studiandone dettagliatamente le prestazioni sia in relazione alla variazione della grandezza del training set, sia al tipo di feature utilizzate.

1 Introduction

Con l'aumento della popolarità e della diffusione del web, e conseguentemente con l'aumento dell'informazione testuale sul web, il processamento automatico di informazione testuale scritta in linguaggio naturale diviene sempre più importante. [3]. A tale fine si usano tecniche di Information Retrieval, che sono alla base degli attuali motori di ricerca web: quando un utente ha un information need, sottomette una query al motore di ricerca, che produrrà come output un insieme di documenti che con buona probabilità dovrebbero contenere l'informazione necessaria all'utente.

1.1 Question Answering

Il Question Answering è una variazione dell'information retrieval (IR): mentre l'IR è orientata ai documenti, il QA ricerca specifiche informazioni all'interno dei documenti cercando di fornire direttamente la risposta all'information need dell'utente [6]. Ed è quindi un compito più difficile rispetto all'IR, poiché fornire una risposta precisa e concisa è più complesso di produrre un intero documento di testo probabilmente molto lungo. Un sistema di QA è generalmente costituito da 4 moduli:

Question Classifier ha il compito di classificare la domanda in categorie prestabilite;

Search Engine basando sulla categoria prodotta dal modulo precedente, ricerca i documenti che hanno probabilità più alta di contenere la risposta;

Text Filter individua le parti di testo all'interno dei documenti, che potrebbero contenere la risposta;

Answer Extractor deduce la risposta dalle parti di testo individuate dal modulo precedente, e le presenta in linguaggio naturale all'utente del sistema.

1.2 Question Classification

Il processo di QC ha quindi il compito di assegnare particolari categorie alle domande, basandosi sul tipo di risposta che la domanda rappresenta. Per classificare le domande, o più in generale del test, è necessario prendere in considerazione due aspetti basilari [2]:

- i tipi delle risposte, le categorie;
- un insieme di regole di classificazione.

1.2.1 Answer Types

Definire un proprio insieme di categorie da utilizzare nella question classification è una delle soluzioni, ma non sempre la migliore. È possibile utilizzare dei sistemi di categorie già usati in precedenza. Il riuso di tali sistemi, oltre a far risparmiare una notevole quantità di tempo, è decisamente utile per comparare i propri risultati con altri ottenuti in precedenza.

I primi sistemi di question classification utilizzavano una suddivisione in un piccolo numero di categorie: sei o sette. Recentemente i ricercatori si sono interessati nella costruzione di categorie migliori: i sistemi attuali solitamente prevedono una divisione in 6-7 categorie a grana grossa, e una successiva suddivisione di ogni categoria in altrettante di dettaglio. Tra i tipi di risposte più famosi troviamo:

- Xin Li e Dan Roth, che propongono una suddivisione in 50 sotto categorie e 6 macro categorie [7]:
  - Abbreviation;
  - Entity;
  - Description;
  - Human;
  - Location;
  - Numerical Value.
- Webclopedia, che usa oltre 140 tipi di risposte, anche chiamati qtargets. Questi sono raggruppati in 8 macro categorie [4]:
  1. relational qtargets;
  2. abstract qtargets;
  3. semantic qtargets;
  4. syntactic qtargets;
  5. role qtargets;
  6. slot qtargets;
  7. lexical qtargets;
  8. combinations of other qtargets.

1.2.2 Classification Strategies

Esistono diverse strategie/regole di classificazione:

Regular expression and hand-written grammar rules sono le prime tecniche utilizzate per la question classification, ma hanno dei grossi limiti, sebbene abbiano avuto successo [2]:

- in primo luogo tali tecniche richiedono molto tempo, poiché sono scritte a mano;
- in secondo luogo sono poco evolvibili, con il cambiare delle categorie;
- se cambiamo l'insieme delle categorie, tutte le regole devono essere riscritte;
- infine sono molto difficili da scrivere, soprattutto quando si usano tante categorie a grana fine.

Non è un caso, infatti, che tali sistemi utilizzino meno di dieci categorie.

Machine Learning Algorithm L'algoritmo più utilizzato in questo campo è SVM, support vector machine. Un'altra architettura molto usata è SNoW [7]. La precision di tali algoritmi, utlizzando solo le parole delle domande, si aggira sul 50%. Ma si possono utilizzare molte altre feature oltre alle semplici parole, come ad esempio: le named entity, head chunk etc ...

Language Modeling Un altro metodo, sempre probabilistico come il machine learning, è il language modeling. Un modello di linguaggio è creato per ogni classe, a partire da tutte le domande appartenenti a quella classe. Dato uno di questi modelli, il nostro obiettivo è scoprire la probabilità con cui la question sia generata da tale modello. Anche in questo caso si possono utilizzare altri insiemi di feature oltre alle semplici parole, come le named entity [8].

2 The starting point

Sebbene sia possibile creare classificatori con regole euristiche costruite ad hoc [9], tale approccio richiede

un'enorme quantità di tempo e di noioso lavoro. Per di più un approccio troppo specifico sarebbe poco flessibile ai cambiamenti che si trovano nel dominio di interesse. Al contrario utilizzando tecniche di machine learning è possibile costruire classificatori che abbiano alte performance e che riescano a gestire migliaia di feature. Oltretutto tale approccio è più flessibile e si adatta facilmente a nuove categorie [10]. Ed è questo l'approccio che intendiamo usare per il nostro classificatore.

Le tecniche di machine learning si distinguono in due grandi categorie:

1. Off-Line;
2. On-Line.

Le principali caratteristiche sulle quali si distinguono le due tipolodie sono: tempo di addestramento, flessibilità e conseguentemente per algoritmo.

Un classificatore off-line deve essere addestrato su un insieme di input (training set). Una volta finita la fase di addestramento, è in grado di classificare, ma non più di imparare. Nel caso si voglia far imparare al classificatore nuovi esempi, l'addestramento deve essere iniziato da zero. Questo porterebbe ad una notevole perdita di tempo se il dominio di inferenza, ma anche le categorie utilizzate, cambiassero nel tempo (rendendo il training set precedentemente utilizzato per "costruire" il classificatore non più coerente con lo stato attuale).

I tempi di training dei classificatori basati su tecniche di machine learning possono essere molto diversi da classificatore a classificatore; tipicamente il tempo di training di un classificatore off-line è maggiore, se confrontato con un classificatore online [5].

Al contrario un algoritmo on-line impara sul campo e durante l'addestramento può comunque essere interrogato, questo consente "all'addestratore" di valutare in continuazione come procede il lavoro di inferenza del dominio da parte del classificatore .

All'apparenza un classificatore on-line offre più vantaggi ma come spesso accade ai "pro" vanno sommati dei "contro", che in questo caso ricadono sulle prestazioni del classificatore; un classificatore on-line è tipicamente peggiore di un classificatore off-line.

2.1 On-Line Algorithm

Adesso che abbiamo una conoscenza generale delle differenze tra le due classi di algoritmi dobbiamo scegliere quale delle due adottare.

Passiamo quindi all'analisi del dominio di interesse, ovvero la question classification (QC); quest'ultima appartiene ad un dominio intrinsecamente variabile, ambiguo e soprattutto dipendente dal tempo.

Il dominio della domande (cioè quello trattato dal question answering e dalla question classification) è variabile nel tempo perché i significati di una parola oltre a essere molteplici possono variare nel tempo, un caso esemplificativo ed evidente è la parola "laico" nella nostra lingua, dove fino a pochissimo tempo fa stava ad indicare la non appartenenza all'ecclesia mentre ora ha in pratica preso il significato di "ateo" e/o "agnostico" (una variazione di significato non indifferente). Poiché il nostro dominio è fortemente influenzato dalle variazioni del contesto nel quale si trova e poiché l'applicabilità degli algoritmi off-line è stata studiata approfonditamente in molti lavori mentre non sono stati approfonditi i comportamenti che hanno gli algoritmi on-line sulla Question Classification, la scelta ricade sugli questi ultimi.

2.2 Winnow

Scelto di utilizzare un algoritmo on-line bisogna scegliere l'algoritmo di base per poi testarlo ed adattarlo alle nostre esigenze.

Tra gli algoritmi on-line la scelta ricade in particolare sull'algoritmo di Winnow.

L'algoritmo di Winnow è composto da un insieme di "esperti" non meglio precisati (che volendo possono essere veri e propri classificatori complessi) ai quali è associato un peso (inizialmente pari a *1*) che vengono interpellati all'arrivo dell'oggetto da classificare.

Ogni "esperto" da come risultato *1* o *0* rispettivamente se l'oggetto appartiene alla classe oppure no. Vengono sommati i risultati ottenuti dal prodotto del peso di ogni classificatore per la predizione corrispondente (*0* oppure *1*). Se il risultato finale supera la soglia (nel caso semplice pari al numero di esperti) al-

lora l'algoritmo generale risponde in modo positivo, altrimenti in modo negativo.

Se l'algoritmo generale fallisce due sono i casi da distinguere:

1. Errore su predizione positiva;
2. Errore su predizione negativa.

Nel primo caso vengono dimezzati i pesi degli "esperti" che hanno predetto *1*, nel secondo invece vengono raddoppiati i pesi degli "esperti" che hanno predetto *1*.

L'algoritmo ha molte qualità tra le quali spicca la capacita di non essere influenzato dalla presenza di molti attributi irrilevanti ("winnow è adatto in teoria per problemi con molti attributi irrilevanti" [10]) e "winnow impara linearmente con il numero di feature rilevanti, e solo logaritmicamente con il numero totale di feature. questa proprietà sembra cruciale in problemi nei quali il numero di feature potenziali è vasto ma solo poche sono rilevanti" [1].

3 Winnow Project

Il nostro classificatore ha la seguente architettura:

Figura 1: Winnow Classifier pipeline

Le domande vengono prese dal file TREC *train_5500.label* disponibile sulla rete, queste vengono "parsate" da un modulo che le da in pasto al secondo modulo che si occupa di trasformare le domande, ora in forma testuale, in forma vettoriale per poi farle analizzare al modulo che poi si occupa di classificare.

3.1 Obiettivi

Entriamo nella fase realizzativa, la piattaforma scelta è quella di WEKA perché ci offre la possibilità di utilizzare e poter paragonare agilmente l'efficacia del classificatore con gli altri già implementati nello strumento. Nel dettaglio gli obiettivi sono:

- Creare un classificatore che rispetti l'interfaccia di WEKA;
- Paragonare il classificatore con algoritmi off-line già implementati in WEKA;
- Analizzare delle prestazioni relativamente a ciascuna feature introdotta;
- Studiare l'andamento del classificatore all'aumento dell'insieme di training.

4 Pre-Processing - Feature Extraction

Per un sistema di classificazione un punto delicato è la scelta delle feature e per tale motivo vanno scelte in modo ponderato e compatibile col dominio che viene trattato dal classificatore.

Il dominio trattato dal classificatore è composto da domande, le quali per loro propria natura sono "compattate" (brevi/compresse) in una certa finestra di lunghezza comunque molto piccola (in media).

L'obiettivo del preprocessing è quello di arricchire le informazioni così da averne poi un beneficio diretto sul processo di classificazione immediatamente successivo.

Le feature che si vogliono introdurre hanno lo scopo di prendere in considerazione:

1. L'ordine delle parole;
2. Il valore semantico delle parole.

La tecnica che si è adottata per soddisfare questi obiettivi è stata quella di aggiungere alla domanda nuovi elementi, ad esempio quelli ottenuti dalla giustapposizione delle parole presenti nelle domanda stessa. Ad

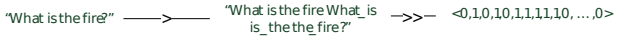

Figura 2: Filters example

esempio se una domanda è nella forma "What is the fire", vengono aggiunte le parole "What_is", "is_the", "the_fire", ottenendo quindi un risultato del genere:

"What is the fire What_is is_the the_fire". È semplice vedere come soprattutto la prima composizione (i.e. "What_is") sia di notevolmente importante nella categorizzazione delle domande.

Con la tecnica sopra descritta sono state aggiunte le seguenti feature:

1. Relazioni tra le parole sia a coppie che a triple;
2. Parole "stemmate" (lasciando anche quelle non originali), nel dettaglio tramite lo "Snowball Stemmer" ed un altro rudimentale sviluppato da noi;
3. Relazione tra la prima parola della domanda e l'ultima;
4. Utilizzo di WordNet per la derivazione del tipo sintattico della parola ed in più la giustapposizione dei tipi sintattici per coppie e per triple.

L'ultimo filtro trasforma la domanda nel vettore delle occorrenze seguendo il modello *Bag of words*.

5 Experimental Set-Up

5.1 Weka Interface

Si è scelto di sviluppare un classificatore che rispettasse l'interfaccia di *WEKA* (v 3.5.7), per poter poi così comparare il lavoro da noi svolto con altri classificatori già implementati all'interno dello strumento.

5.2 Filters

Per aggiungere feature alla domanda sono stati utilizzati:

- SnowballStemmer - la cui interfaccia è esposta da WEKA;
- WordNet 3 - per la derivazione dei sinonimi;
- Più quelli da noi creati (illustrati nella sezione precedente).

5.3 Report

Per avere una visione più intuitiva delle prestazioni del classificatore sono stati creati dei report grafici creati con *JFreechart 1.0.9*.

Tale strumento consente per l'appunto di poter creare dei report riassuntivi e comparativi partendo da dati numerici.

5.4 Winnow Classifier The Algorithm

L'algoritmo di Winnow realizzato ha la seguente architettura: ad ogni "classe" da classificare è associata una collezione di così detti "esperti", ogni esperto è relativo ad una parola, ovvero un esperto viene interpellato ogni qual volta la parola associata ad esso è presente nella domanda. L'algoritmo quindi ha come input una domanda, per ogni parola presente nella domanda interroga il relativo esperto, se la somma dei punteggi di tutti gli esperti supera una certa soglia allora la predizione per quella determinata classe è positiva altrimenti no. Ogni esperto quindi associa un punteggio (o peso) alla parola al quale è associato, tale punteggio viene gestito secondo l'algoritmo classico, cioè incrementandolo (raddoppiando il peso) in caso di errore su predizione negativa o decrementandolo (dimezzando il peso) in caso di errore su predizione positiva.

5.5 SMO Comparison

Tra i classificatori off-line con cui effettuare la comparazione è stato scelto il classificatore SMO, integrato in WEKA.

Per poter effettuare la comparazione con un classificatore di tipo profondamente diverso (SMO è off-line al contrario del nostro) è stato necessario addestrare diverse volte il classificatore, nel dettaglio ad ogni cambiamento dell'insieme di training.

5.6 Hardware

La macchina di test ha le seguenti caratteristiche principali:

- Processore: Intel Core 2 Duo T7500;

- RAM: 3GB;
- S.O.: Windows Vista Business x64 SP1.

6 Experimental Result

Analizziamo ora i risultati ottenuti. Ci concentriamo dapprima sul nostro classificatore, dopodichè sul confronto con SMO.

6.1 Winnow Result

In questo paragrafo analizziamo, quindi, alcuni risultati ottenuti dal nostro classificatore a seconda delle feature utilizzate.

6.1.1 All Feature (best result)

Il miglior risultato ottenuto dal nostro classificatore, utilizzando tutte le feature descritte nei paragrafi precedenti (dalle parole stemmate all'ordine delle parole nella domanda), ha portato ad un'accuracy dell'80.6%. Riportiamo di seguito uno schema riassuntivo del risultato ottenuto nella stessa forma che prevede l'Evaluator di Weka:

```
Correctly Classified Instances    403         80.6    %
Incorrectly Classified Instances   71         14.2    %
Kappa statistic                     0.8098
Mean absolute error                 0.065
Root mean squared error             0.255
Relative absolute error            25.9565 %
Root relative squared error        72.1325 %
UnClassified Instances             26          5.2    %
Total Number of Instances         500
```

Volendo però analizzare più nel dettaglio tale risultato, ci accorgiamo come il classificatore non è performante alla stessa maniera in tutte le categorie. Non capita mai che il sistema classifichi erroneamente una domanda appartentente alla macro categoria abbreviation. Mentre esattamente opposto è il comportamento relativo alla categoria entity, che è spesso causa di errore.

```
  a   b   c   d   e   f   <-- classified as
  7   2   0   0   0   0 |   a = ABBR
  0 129   2   0   0   6 |   b = DESC
  0  10  58   0   4   7 |   c = ENTY
  0   0   3  56   2   0 |   d = HUM
  0   2  13   0  59   3 |   e = LOC
  0   4   6   5   2  94 |   f = NUM
```

Ma un analisi del genere è ancora sommaria: la precision non è l'unico parametro con cui valutare le prestazioni di un classificatore. Nello studio dei classificatori i parametri più utilizzati sono infatti la precision e la recall. Il primo si riferisce al rapporto tra il numero di domande classificate correttamente per una certa categoria diviso il numero di domande classificate in totale sempre appartenenti alla stessa categoria. Il secondo, invece, corrisponde al rapporto tra il numero di domande classificate correttamente per una categoria diviso tutte le domande appartenenti a tale categoria.

Difatti, analizzando nuovamente i dati, possiamo notare come per la categoria abbreviation la precision è 1, ma non la recall. Mentre per la classe entity sia la precision che la recall sono tra i valori più bassi fatti registrare dal nostro classificatore. La tabella 1 riporta i dati tecnici di cui stiamo parlando, più altri che sicuramente saranno noti agli esperti del settore e di Weka.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.778	0	1	0.778	0.875	0.889	ABBR
0.942	0.053	0.878	0.942	0.908	0.934	DESC
0.734	0.061	0.707	0.734	0.72	0.749	ENTY
0.918	0.012	0.918	0.918	0.918	0.925	HUM
0.766	0.02	0.881	0.766	0.819	0.893	LOC
0.847	0.044	0.855	0.847	0.851	0.947	NUM

Tabella 1: Detailed Accuracy By Class

Infine, come ci eravamo proposti, analizziamo anche l'andamento delle prestazioni del classificatore al variare dell'insieme di input, questo per capire come si comporta l'algoritmo, che, ripetiamo, essendo online, è in grado di imparare anche durante l'utilizzo dello stesso. Si può osservare tale andamento nella figura 3.

Si noti come il comportamento di winnow sia piuttosto oscillatorio e come, nonostante una breve fase iniziale di assestamento, il comportamento di winnow, seppur restando oscillatorio, si stabilizza intorno a certi valori, senza mostrare un incremento notevole all'aumentare dell'input. Tale fenomeno ovviamente potrebbe dipendere sia dal training set, che sebbene comprende 5500 domande, potrebbe ritenersi ancora piuttosto ridotto; sia dalla scelta delle

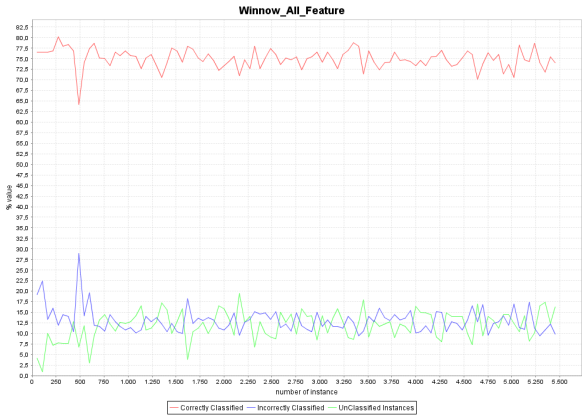

Figura 3: Andamento di winnow con tutte le feature

feature utilizzate; ma anche dai parametri/pesi propri dell'algoritmo che, con un'accurata fase di tuning, potrebbero portare a dei miglioramenti sensibili delle performance.

6.1.2 Zero feature

Mostriamo ora, invece, i risultati che winnow otteneva prima di applicare qualsiasi feature da noi proposta. Ossia in questo esempio viene utilizzato semplicemente l'approccio bag of word, dove ogni esperto è una delle parole che occorrono nella domanda, senza andare a fare nessuna modifica di quest'ultima. Si può notare come l'accuracy sia intorno al 50% e quindi come l'utilizzo delle feature da noi studiate, abbia portato ad un miglioramento del 30%.

```
Correctly Classified Instances     253        50.6     %
Incorrectly Classified Instances   101        20.2     %
Kappa statistic                      0.6453
Mean absolute error                  0.0927
Root mean squared error              0.3045
Relative absolute error             48.7635 %
Root relative squared error         98.0838 %
UnClassified Instances             146        29.2     %
Total Number of Instances          500
```

Dalla tabella 2 si può notare come la classe che in queste condizioni ha il risultato peggiore, sia in termini di precision che di recall, è la classe description, che è quindi quella che trae maggior beneficio dalle feature da noi utilizzate.

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.875	0.009	0.7	0.875	0.778	0.886	ABBR
0.585	0.083	0.554	0.585	0.569	0.581	DESC
0.701	0.167	0.495	0.701	0.58	0.71	ENTY
0.8	0.027	0.846	0.8	0.822	0.871	HUM
0.75	0.049	0.785	0.75	0.767	0.833	LOC
0.709	0.012	0.961	0.709	0.816	0.91	NUM

Tabella 2: Zero feature result

6.1.3 Stemmer only

Un altro risultato degno di nota, è quello prodotto dall'utilizzo delle sole parole stemmate. Quest'unica feature da sola apporta un miglioramento del 15% rispetto al caso di nessuna feature utilizzata. Riteniamo quindi tale caratteristica di estrema importanza nel dominio della question classification.

```
Correctly Classified Instances     325        65       %
Incorrectly Classified Instances   141        28.2     %
Kappa statistic                      0.6123
Mean absolute error                  0.128
Root mean squared error              0.3578
Relative absolute error             51.9759 %
Root relative squared error        102.0796 %
UnClassified Instances              34         6.8     %
Total Number of Instances          500
```

6.2 Winnow vs SMO

In questo paragrafo compariamo invece il risultato prodotto da winnow, un algoritmo on-line, con SMO, un'implementazione particolare delle Support Vector Machine, ossia un algoritmo off-line.

6.2.1 All Feature

In questo test sono state utilizzate tutte le feature da noi analizzate: notiamo nella figura 4 come SMO abbia prestazioni superiori rispetto a Winnow, ad esclusione di una fase transitoria iniziale. Ma già dopo un training set composto da 1000 domande, SMO è decisamente supeirore a Winnow. Oltretutto SMO mostra una migliore tendenza a crescere in prestazione con l'aumentare dell'insieme di training, fino ad arrivare a circa il 90% di accuracy, mentre Winnow resta piuttosto costante. Però è doveroso dire che SMO, a differenza di Winnow, oltre a essere molto più lento nella fase di addestramento, non è più in grado di imparare dopo che tale fase sia terminata.

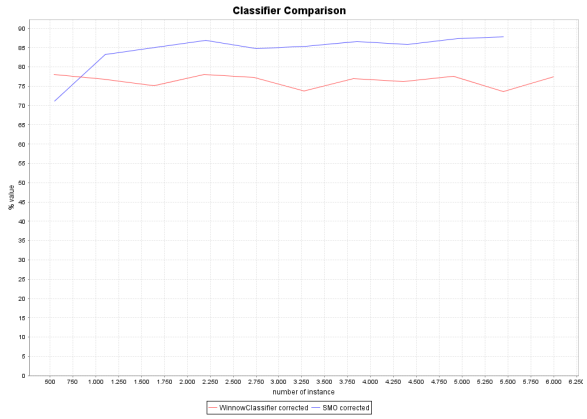

Figura 4: Winnow vs SMO with all feature

6.2.2 Zero Feature

In questo secondo test, rappresentato dalla figura 5, non viene utilizzata alcuna feature, ma viene semplicemente data in pasto al classificatore la domanda così com'è. Ciò che risulta evidente è come il compostamento di SMO sia molto simile a quello del caso precedente (con tutte le feature), mentre Winnow mostra delle serie limitazioni qualora gli esperti utilizzati non sia poi così esperti.

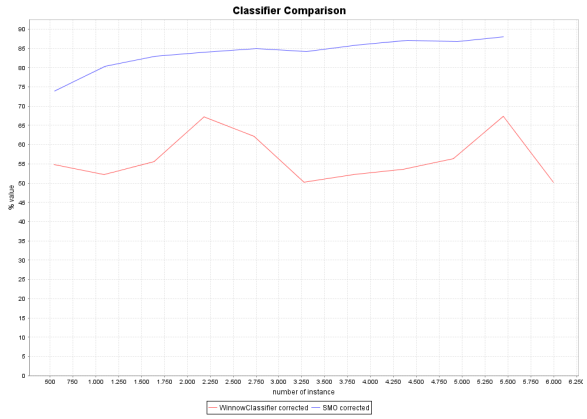

Figura 5: Winnow vs SMO with zero feature

7 Conclusion

In conclusione si può dire che l'algoritmo di Winnow non tradisce le aspettative, pur avendo prestazioni nettamente inferiori ad un classificatore off-line (ossia SMO).

Un'interessante caratteristica che si può notare di Winnow è come il suo comportamento degradi molto poco con l'aumentare del numero di esperti/feature utilizzate, mentre anche solo pochi esperti rilevanti apportano un grande contributo alle performance del classificatore. Questo va a conferma di quanto già è stato notato da [10] e da [1].

A tal proposito, le feature che hanno contribuito maggiormente all'incremento delle prestazioni sono lo stemmer e la coppia formata dalle prime due parole di ogni domanda.

Abbiamo, poi, osservato come la classe entity sia quella di più difficile classificazione, e che quindi richiederebbe l'utilizzo di feature adatte e specifiche allo scopo, come le tecniche di named entity recognition, con l'uso di opportuni tag da inserire/sostituire all'interno della domanda.

Un'altra interessante osservazione, come visibile dai grafici, è il comportamento oscillatorio di Winnow, caratteristica propria dell'algoritmo. L'ampiezza dell'oscillazione dipende dai parametri interni dell'algoritmo.

Considerando che le feature introdotte sono semplici e che il tuning non è stato fatto in modo minuzioso, i margini di miglioramento di certo ci sono.

Riferimenti bibliografici

[1] Jeff L. Rosen Andrew J. Carlson, Chad M. Cumby and Dan Roth. Snow user's guide. `http://l2r.cs.uiuc.edu/~danr/Papers/userguide.ps.gz`, 1999.

[2] Jonathan Brown. Entity-tagged language models for question classification in a qa system. `http://nyc.lti.cs.cmu.edu/IRLab/11-743s04/jonbrown/Brown-IRLab.pdf`, 2004.

[3] Zhalaing Cheung, Khanh Linh Phan, Ashesh Mahidadia, and Achim Hoffmann. Fea-

ture extraction for learning to classify questions. http://ciir.cs.umass.edu/~weili/papers/qc.pdf.

[4] Laurie Gerber, Ulf Hermjakob, Eduard Hovy, and Deepak Ravichandran. The isi question answer typology, qtargets used in webclopedia. http://www.isi.edu/natural-language/projects/webclopedia/Taxonomy/taxonomy_toplevel.html, 2002.

[5] Mona Soliman Habib. Addressing scalability issues of named entity recognition using multi-class support vector machines, 2006.

[6] Wei Li. Question classification using language modeling. http://ciir.cs.umass.edu/~weili/papers/qc.pdf, 2003.

[7] Xin Li and Dan Roth. Learning question classifiers. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.7195&rep=rep1&type=pdf, 2002.

[8] D. Pinto, W. Croft, M. Branstein, R. Coleman, M. King, W.Li, and X. Wei. Quasm: A system for question answering using semi-structured data, 2002.

[9] Ellen M. Voorhees. Overview of trec 2001. In *TREC*, 2001.

[10] Tong Zhang. Large margin winnow methods for text categorization. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=398E23634B77B6D88BECA4A8F1BE7FEA?doi=10.1.1.36.6312&rep=rep1&type=pdf, 2000.