Soft Cardinality + ML: Learning Adaptive Similarity Functions for Cross-lingual Textual Entailment

Sergio Jimenez
Universidad Nacional
de Colombia, Bogota,
Ciudad Universitaria
edificio 453, oficina 220
sgjimenezv@unal.edu.co

Claudia Becerra
Universidad Nacional
de Colombia, Bogota
cjbecerrac@unal.edu.co

Alexander Gelbukh
CIC-IPN
Av. Juan Dios Bátiz,
Av. Mendizábal, Col.
Nueva Industrial Vallejo,
CP 07738, DF, México
gelbukh@gelbukh.com

Abstract

This paper presents a novel approach for building adaptive similarity functions based on cardinality using machine learning. Unlike current approaches that build feature sets using similarity scores, we have developed these feature sets with the cardinalities of the commonalities and differences between pairs of objects being compared. This approach allows the machine-learning algorithm to obtain an asymmetric similarity function suitable for directional judgments. Besides using the classic set cardinality, we used soft cardinality to allow flexibility in the comparison between words. Our approach used only the information from the surface of the text, a stop-word remover and a stemmer to address the cross-lingual textual entailment task 8 at SEMEVAL 2012. We have the third best result among the 29 systems submitted by 10 teams. Additionally, this paper presents better results compared with the best official score.

1 Introduction

Adaptive similarity functions are those functions that, beyond using the information of two objects being compared, use information from a broader set of objects (Bilenko and Mooney, 2003). Therefore, the same similarity function may return different results for the same pair of objects, depending on the context of where the objects are. Adaptability is intended to improve the performance of the similarity function in relation to the task in question associated with the entire set of objects. For example, adaptiveness improves relevance of documents retrieved for a query in an information retrieval task for a particular document collection.

In text applications there are mainly three methods to provide adaptiveness to similarity functions: term weighting, adjustment or learning the parameters of the similarity function, and machine learning. Term weighting is a common practice that assigns a degree of importance to each occurrence of a term in a text collection (Salton and Buckley, 1988; Lan et al., 2005). Secondly, if a similarity function has parameters, these can be adjusted or learned to adapt to a particular data set. Depending on the size of the search space defined by these parameters, they can be adjusted either manually or using a technique of AI. For instance, Jimenez et al. manually adjusted a single parameter in the generalized measure of Monge-Elkan (1996) (Jimenez et al., 2009) and Ristrad and Yanilios (1998) learned the costs of editing operations between particular characters for the Levenshtein distance (1966) using HMMs. Thirdly, the machine-learning approach aims to learn a similarity function based on a vector representation of texts using a subset of texts for training and a learning function (Bilenko and Mooney, 2003). The three methods of adaptability can also be used in a variety of combinations, e.g. term weighting in combination with machine learning (Debole and Sebastiani, 2003; Lan et al., 2005). Finally, to achieve adaptability, other approaches use data sets considerably larger, such as large corpora or the Web, e.g. distributional similarity (Lee, 1999).

In the machine-learning approach, a vector representation of texts is used in conjunction with an algorithm of classification or regression (Alpaydin, 2004). Each vector of features $\langle f_1, f_2, \ldots, f_m \rangle$ is associated to each pair $\langle T_i, T_j \rangle$ of texts. Thus, Bilenko et al. (2003) proposed a set of features indexed by the data set vocabulary, similar to Zanzotto et al., (2009) who used fragments of parse trees. However, a more common approach is to select as features the scores of different similarity functions. Using these features, the machine-learning algorithm discovers the relative importance of each feature and a combination mechanism that maximizes the alignment of the final result with a gold standard for the particular task.

In this paper, we propose a novel approach to extract feature sets for a machine-learning algorithm using car-

*First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 684–688,
Montréal, Canada, June 7-8, 2012.

dinalities rather than scores of similarity functions. For instance, instead of using as a feature the score obtained by the Dice's coefficient (i.e. $2\times|T_i\cap T_j|/|T_i|+|T_j|$), we use $|T_i|$, $|T_j|$ and $|T_i \cap T_j|$ as features. The rationale behind this idea is that despite the similarity scores being suitable for learning a combined function of similarity, they hide the information imbalance between the original pair of texts. Our hypothesis is that the information coded in this imbalance could provide the machine-learning algorithm with better information to generate a combined similarity score. For instance, consider these pairs of texts: ⟨ "*The beach house is white*.", "*The house was completely empty*."⟩ and ⟨ "*The house*", "*The beach house was completely empty and isolated*"⟩. Both pairs have the same similarity score using the Dice coefficient, but it is evident that the latter has an imbalance of information lost in that single score. This imbalance of information is even more important if the task requires to identify directional similarities, such as "T_1 is more similar to T_2, than T_2 is to T_1".

However, unlike the similarity functions, which are numerous, there is only one set cardinality. This issue can be addressed using the soft cardinality proposed by Jimenez et al. (2010), which uses an auxiliary function of similarity between elements to make a soft count of the elements in a set. For instance, the classic cardinality of the set $A = \{$ "*Sunday*", "*Saturday*" $\}$ is $|A| = 2$; and the soft cardinality of the same set, using a normalized edit-distance as auxiliary similarity function, is $|A|'_{sim} = 1.23$ because of the commonalities between both words. Furthermore, soft cardinality allows weighting of elements giving it additional capacity to adapt.

We used the proposed approach to participate in the cross-lingual textual-entailment task 8 at SEMEVAL 2012. The task was to recognize bidirectional, forward, backward or lack of entailment in pairs of texts written in five languages. We built a system based on the proposed method and the use of surface information of the text, a stop-word remover and a stemmer. Our system achieved the third best result in official classification and, after some debugging, we are reporting better results than the best official scores.

This paper is structured as follows. Section 2 briefly describes soft cardinality and other cardinalities for text applications. Section 3 presents the proposed method. Experimental validation is presented in Section 4. A brief discussion is presented in Section 5. Finally, conclusions are drawn in Section 6.

2 Cardinalities for text

Cardinality is a measure of counting the number of elements in a set. The cardinality of classical set theory represents the number of non-repeated elements in a set. However, this cardinality is rigid because it counts in the same manner very similar or highly differentiated elements. In text applications, text can be modeled as a set of words and a desirable cardinality function should take into account the similarities between words. In this section, we present some methods to soften the classical concept of cardinality.

2.1 Lemmatizer Cardinality

The simplest approach is to use a stemmer that collapses words with common roots in a single lemma. Consider the sentence: "*I loved, I am loving and I will love you*". The plain word counting of this sentence is 10 words. The classical cardinality collapses the three occurrences of the pronoun "*I*" giving a count of 8. However, a lemmatizer such as Porter's stemmer (1980) also collapses the words "*loved*", "*loving*" and "*love*" in a single lemma "*love*" for a count of 6. Thus, when a text is lemmatized, it induces a relaxation of the classical cardinality of a text. In addition, to provide corpus adaptability, a weighted version of this cardinality can add weights associated with each word occurrence instead of adding 1 for each word (e.g. tf-idf).

2.2 LCS cardinality

Longest common subsequence (LCS) length is a measure of the commonalities between two texts, unlike set intersection, taking into account the order. Therefore, a cardinality function of a pair of texts A and B could be $|A \cap B| = len(LCS(A, B))$, $|A| = len(A)$ and $|B| = len(B)$. Functions $len(*)$ and $LCS(*, *)$ calculate length and LCS respectively, either in character or word granularity.

2.3 Soft Cardinality

Soft cardinality is a function that uses an auxiliary similarity function to make a soft count of the elements (i.e. words) in a set (i.e. text) (Jimenez et al., 2010). The auxiliary similarity function can be any measure or metric that returns scores in the interval $[0, 1]$, with 0 being the lowest degree of similarity and 1 the highest (i.e. identical words). Clearly, if the auxiliary similarity function is a rigid comparator that returns 1 for identical words and 0 otherwise, the soft cardinality becomes the classic set cardinality.

The soft cardinality of a set $A = \{a_1, a_2, \ldots, a_{|A|}\}$ can be calculated by the following expression: $|A|'_{sim} \simeq \sum_i^{|A|} w_{a_i} \left(\sum_j^{|A|} sim(a_i, a_j)^p\right)^{-1}$. Where $sim(*, *)$ is the auxiliary similarity function for approximate word comparison, w_{a_i} are weights associated with each word a_i, and p is a tuning parameter that controls the degree of smoothness of the cardinality, i.e. if $0 \leftarrow p$ all elements in a set are considered identical and if $p \rightarrow \infty$ soft cardinality becomes classic cardinality.

2.4 Dot-product VSM "Cardinality"

Resemblance coefficients are cardinality-based similarity functions. For instance, the Dice coefficient is the ratio between the cardinality of the intersection divided by the arithmetic mean of individual cardinalities: $2\times|A\cap B|/|A|+|B|$. The cosine coefficient is similar but instead of using the arithmetic mean it uses the geometric mean: $|A\cap B|/\sqrt{|A|}\times\sqrt{|B|}$. Furthermore, the cosine similarity is a well known metric used in the vector space model (VSM) proposed by Salton et al. (1975) $cosine(A,B) = \frac{\sum w_{a_i}\times w_{b_i}}{\sqrt{\sum w_{a_i}^2}\times\sqrt{\sum w_{b_i}^2}}$. Clearly, this expression can be compared with the cosine coefficient interpreting the dot-product operation in the cosine similarity as a cardinality. Thus, the obtained cardinalities are: $|A\cap B|_{vsm} = \sum w_{a_i}^p \times w_{b_i}^p$, $|A|_{vsm} = \sum w_{a_i}^{2p}$ and $|B|_{vsm} = \sum w_{b_i}^{2p}$. The exponent p controls the effect of weighting providing no effect if $0 \leftarrow p$ or emphasising the weights if $p > 0$. In a similar application, Gonzalez and Caicedo (2011) used $p = 0.5$ and normalization justified by the quantum information retrieval theory.

3 Learning Similarity Functions from Cardinalities

Different similarity measures use different knowledge, identify different types of commonalities, and compare objects with different granularity. In many of the automatic text-processing applications, the qualities of several similarity functions may be required to achieve the final task. The combination of similarity scores with a machine-learning algorithm to obtain a unified effect for a particular task is a common practice (Bilenko et al., 2003; Malakasiotis and Androutsopoulos, 2007; Malakasiotis, 2009). For each pair of texts for comparison, there is provided a vector representation based on multiple similarity scores as a set of features. In addition, a class attribute is associated with each vector which contains the objective of the task or the gold standard to be learned by the machine-learning algorithm.

However, the similarity scores conceal important information when the task requires dealing with directional problems, i.e. whenever the order of comparing each pair of texts is related with the class attribute. For instance, textual entailment is a directional task since it is necessary to recognize whether the first text entails the second text or vice versa. This problem can be addressed using asymmetric similarity functions and including scores for $sim(A,B)$ and $sim(B,A)$ in the resulting vector for each pair $\langle A,B\rangle$. Nevertheless, the similarity measures that are more commonly used are symmetric, e.g. edit-distance (Levenshtein, 1966), LCS (Hirschberg, 1977), cosine similarity, and many of the current semantic relatedness measures (Pedersen et al., 2004). Although, there are asymmetric measures such as the Monge-Elkan measure (1996) and the measure proposed by Corley and Mihalcea (Corley and Mihalcea, 2005), they are outnumbered by the symmetric measures. Clearly, this situation restricts the use of the machine learning as a method of combination for directional problems.

Alternatively, we propose the construction of a vector for each pair of texts using cardinalities instead of similarity scores. Moreover, using cardinalities rather than similarity scores allows the machine-learning algorithm to discover patterns to cope with directional tasks.

Basically, we propose to use a set with six features for each cardinality function: $|A|$, $|B|$, $|A\cap B|$, $|A\cup B|$, $|A-B|$ and $|B-A|$.

4 Experimental Setup

4.1 Cross-lingual Textual Entailment (CLTE) Task

This task consist of recognizing in a pair of topically related text fragments T_1 and T_2 in different languages, one of the following possible entailment relations: i) *bidirectional* $T_1 \Rightarrow T_2 \wedge T_1 \Leftarrow T_2$, i.e. semantic equivalence; ii) *forward* $T_1 \Rightarrow T_2 \wedge T_1 \not\Leftarrow T_2$; iii) *backward* $T_1 \not\Rightarrow T_2 \wedge T_1 \Leftarrow T_2$; and iv) *no entailment* $T_1 \not\Rightarrow T_2 \wedge T_1 \not\Leftarrow T_2$. Besides, both T_1 and T_2 are assumed to be true statements; hence contradictory pairs are not allowed.

Data sets consist of a collection of 1,000 text pairs (500 for training and 500 for testing) each one labeled with one of the possible entailment types. Four balanced data sets were provided using the following language pairs: German-English (deu-eng), French-English (fra-eng), Italian-English (ita-eng) and Spanish-English (spa-eng). The evaluation measure for experiments was accuracy, i.e. the ratio of correctly predicted pairs by the total number of predictions. For a comprehensive description of the task see (Negri et al., 2012).

4.2 Experiments

Given that each pair of texts $\langle T_1,T_2\rangle$ are in different languages, a pair of translations $\langle T_1^t,T_2^t\rangle$ were provided using Google Translate service. Thus, each one of the text pairs $\langle T_1,T_2^t\rangle$ and $\langle T_1^t,T_2\rangle$ were in the same language. Then, all produced pairs were pre-processed by removing stop-words in their respective languages. Finally, all texts were lemmatized using Porter's stemmer (1980) for English and Snowball stemmers for other languages using an implementation provided by the NLTK (Loper and Bird, 2002).

Then, different set of features were generated using similarity scores or cardinalities. While each symmetric similarity function generates 2 features i)$sim(T_1,T_2^t)$ and ii)$sim(T_1^t,T_2)$, asymmetric functions generate two additional features iii)$sim(T_2^t,T_1)$ and iv)$sim(T_2,T_1^t)$.

On the other hand, each cardinality function generates 12 features: i) $|T_1|$, ii) $|T_2^t|$, iii) $|T_1 \cap T_2^t|$, iv) $|T_1 \cup T_2^t|$, v) $|T_1 - T_2^t|$, vi) $|T_2^t - T_1|$, vii) $|T_1^t|$, viii) $|T_2|$, ix) $|T_1^t \cap T_2|$, x) $|T_1^t \cup T_2|$, xi) $|T_1^t - T_2|$, and xii) $|T_2 - T_1^t|$. Various combinations of cardinalities, symmetric and asymmetric functions were used to generate the following feature sets:

Sym.simScores: scores of the following symmetric similarity functions: Jaccard, Dice, and cosine coefficients using classical cardinality and soft cardinality (edit-distance as auxiliar sim. function). In addition, cosine similarity, softTFIDF (Cohen et al., 2003) and edit-distance (total 18 features).

Asym.LCS.sim: scores of the following asymmetric similarity functions: $sim(T_1, T_2) = {}^{lcs(T_1,T_2)}/_{len(T_1)}$ and $sim(T_1, T_2) = {}^{lcs(T_1,T_2)}/_{len(T_2)}$ at character level (4 features).

Classic.card: cardinalities using classical set cardinality (12 features).

Dot.card.w: dot-product cardinality using idf weights as described in Section 2.4, using $p = 1$ (12 features).

LCS.card: LCS cardinality at word-level using idf weights as described in Section 2.1 (12 features).

SimScores: combined features sets from Sym.SimScores, Asym.LCS.sim and the generalized Monge-Elkan measure (Jimenez et al., 2009) using $p = 1, 2, 3$ (30 features).

Dot.card.w.0.5: same as Dot.card.w using $p = 0.5$.

Classic.card.w: classical cardinality using idf weights (12 features).

Soft.card.w: soft cardinality using idf weights as described in Section 2.3 using $p = 1, 2, 3, 4, 5$ (60 features).

The machine-learning classification algorithm for all feature sets was SVM (Cortes and Vapnik, 1995) with the complexity parameter $C = 1.5$ and a linear polynomial kernel. All experiments were conducted using WEKA (Hall et al., 2009).

4.3 Results

In Semeval 2012 exercise, participants were given a particular subdivision into training and test subsets for each data set. For official results, participants received only the gold-standard labels for the subset of training, and accuracies of each system in the test subset was measured by the organizers. In Table 1, the results for that particular division are shown. At the bottom of that table, the official results for the first three systems are shown. Our system, "3rd.Softcard" was configured using soft cardinality with edit-distance as auxiliary similarity function and $p = 2$. Erroneously, at the time of the submission, all texts in the 5 languages were lemmatized using an English stemmer and stop-words in all languages were aggregated into a single set before the withdrawal. In spite of these bugs, our system was the third best score.

FEATURES	SPA	ITA	FRA	DEU	avg.
Sym.simScores	0.404	0.410	0.410	0.410	0.409
Asym.LCS.sim	0.490	0.492	0.482	0.474	0.485
Classic.card	0.560	0.534	0.570	0.542	0.552
Dot.card.w	0.562	0.568	0.550	0.548	0.557
LCS.card	0.606	0.566	0.568	0.558	0.575
SimScores	0.600	0.562	0.568	0.572	0.576
Dot.card.w.0.5	0.584	0.574	0.586	0.572	0.579
Classic.card.w	0.584	0.576	0.588	0.590	0.585
Soft.card.w	0.598	**0.602**	**0.624**	**0.604**	**0.607**
SEMEVAL 2012 OFFICIAL RESULTS					
1st.HDU.run2	**0.632**	0.562	0.570	0.552	0.579
2nd.HDU.run1	0.630	0.554	0.564	0.558	0.577
3rd.Softcard	0.552	0.566	0.570	0.550	0.560

Table 1: Accuracy results for Semeval2012 task 8

Soft.card.w	**60.174(1.917)%**	imprv.	Sign.
Sym.simScore	39.802(1.783)%	51.2%	<0.001
Asym.LCS.sim	48.669(1.820)%	23.6%	<0.001
Classic.card	55.278(2.422)%	8.9%	0.010
Dot.card.w	54.906(2.024)%	9.6%	0.004
LCS.card	55.131(2.471) %	9.1%	0.015
SimScores	56.889(2.412) %	5.8%	0.124
Dot.card.w.0.5	57.114(2.141)%	5.4%	0.059
Classic.card.w	56.708(2.008)%	6.1%	0.017

Table 2: Average accuracy comparison vs. Soft.card.w in 100 runs

To compare our approach of using feature sets based on soft cardinality versus other approaches, we generated 100 random training-test subdivisions (50%-50%) of each data set. The average results were compared and tested statistically with the paired T-tested corrected test. Results, deviations, the percentage of improvement, and its significance in comparison with the Soft.card.w system are shown in Table2.

5 Discusion

Results in Table 2 show that our hypothesis that feature sets obtained from cardinalities should outperform features sets obtained from similarity scores was demostrated when compared versus similarity functions alternatively symmetrical or asymetrical. However, when our approach is compared with a feature set obtained by combining symmetric and asymmetric functions, we obtained an improvement of 5.8% but only with a significance of 0.124. Regarding soft cardinality compared to alternative cardinalities, soft cardinality outperformed others in all cases with significance <0.059.

6 Conclusions

We have proposed a new method to compose feature sets using cardinalities rather than similarity scores. Our approach proved to be effective for directional text comparison tasks such as textual entailment. Furthermore, the soft cardinality function proved to be the best for obtaining such sets of features.

Acknowledgments

This research was funded by the Systems and Industrial Engineering Department, the Office of Student Welfare of the National University of Colombia, Bogotá, and throught a grant from the Colombian Department for Science, Technology and Innovation Colciencias, proj. 110152128465. The second author recognizes the support from Mexican Government (SNI, COFAA-IPN, SIP 20113295, CONACYT 50206-H) and CONACYT–DST India (proj. "Answer Validation through Textual Entailment").

References

Ethem Alpaydin. 2004. *Introduction to Machine Learning*. MIT press.

Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington, D.C.

Mikhail Bilenko, Raymond Mooney, William Cohen, Pradeep Ravikumar, and Stephen Fienberg. 2003. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.

William W Cohen, Pradeep Ravikumar, and Stephen E Fienberg. 2003. A comparison of string distance metrics for Name-Matching tasks. In *Proc. of the IJCAI2003 Workshop on Information Integration on the Web II Web03*.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Stroudsburg, PA.

Corinna Cortes and Vladimir N. Vapnik. 1995. Support-Vector networks. *Machine Learning*, 20(3):273–297.

Franca Debole and Fabrizio Sebastiani. 2003. Supervised term weighting for automated text categorization. In *Proc. of the 2003 ACM symposium on applied computing*, New York, NY.

Fabio A. Gonzalez and Juan C. Caicedo. 2011. Quantum latent semantic analysis. In *Proc. of the Third international conference on Advances in information retrieval theory*.

Mark Hall, Frank Eibe, Geoffrey Holmes, and Bernhard Pfahringer. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18.

Daniel S. Hirschberg. 1977. Algorithms for the longest common subsequence problem. *J. ACM*, 24(4):664–675.

Sergio Jimenez, Claudia Becerra, Alexander Gelbukh, and Fabio Gonzalez. 2009. Generalized Monge-Elkan method for approximate text string comparison. In *Computational Linguistics and Intelligent Text Processing*, volume 5449 of *LNCS*, pages 559–570.

Sergio Jimenez, Fabio Gonzalez, and Alexander Gelbukh. 2010. Text comparison using soft cardinality. In *String Processing and Information Retrieval*, volume 6393 of *LNCS*, pages 297–302.

Man Lan, Chew-Lim Tan, Hwee-Boon Low, and Sam-Yuan Sung. 2005. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, New York, NY.

Lillian Lee. 1999. Measures of distributional similarity. In *Proc. of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, College Park, Maryland.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, Philadelphia, PA.

Prodromos Malakasiotis and Ion Androutsopoulos. 2007. Learning textual entailment using SVMs and string similarity measures. In *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, Stroudsburg, PA.

Prodromos Malakasiotis. 2009. Paraphrase recognition using machine learning to combine similarity measures. In *Proc. of the ACL-IJCNLP 2009 Student Research Workshop*, Stroudsburg, PA.

Alvaro E. Monge and Charles Elkan. 1996. The field matching problem: Algorithms and applications. In *Proc. KDD-96*, Portland, OR.

Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. 2012. 2012. semeval-2012 task 8: Cross-lingual textual entailment for content synchronization. In *In Proc. of the 6th International Workshop on Semantic Evaluation (SemEval 2012)*, Montreal, Canada.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity: measuring the relatedness of concepts. In *Proc. HLT-NAACL–Demonstration Papers*, Stroudsburg, PA.

Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 3(14):130–137.

Eric S. Ristad and Peter N. Yianilos. 1998. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Gerard Salton, Andrew K. C. Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Natural Language Engineering*, 15(Special Issue 04):551–582.