STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 3, 2010

ALIGNMENT OF CUSTOM STANDARDS BY MACHINE LEARNING ALGORITHMS

ADELA SÎRBU[1,2], LAURA DIOŞAN[1,2],
ALEXANDRINA ROGOZAN[1], AND JEAN-PIERRE PÉCUCHET[1]

Abstract. Building an efficient model for automatic alignment of terminologies would bring a significant improvement to the information retrieval process. We have developed and compared two machine learning based algorithms whose aim is to align 2 custom standards built on a 3 level taxonomy, using kNN and SVM classifiers that work on a vector representation consisting of several similarity measures. The weights utilized by the kNN were optimized with an evolutionary algorithm, while the SVM classifier's hyper-parameters were optimized with a grid search algorithm. The database used for train was semi automatically obtained by using the Coma++ tool. The performance of our aligners is shown by the results obtained on the test set.

1. Introduction

The need for terminology integration has been widely recognized in different areas (economical, custom etc) leading to a number of efforts for defining standardized and complete terminologies. It is acknowledged by literature that the creation of a single universal terminology for a particular domain is neither possible nor beneficial because different tasks and viewpoints require different, often incompatible conceptual choices. Considering these aspects, an important research direction is searching for an automatic recognition of concepts with the same meaning, even though they have not the same syntactic representation.

The main aim of our work is to design an algorithm in order to perform alignments between two terminologies. The goal of the alignments is to put in correspondence concepts that refer to the same thing, but which appear under different forms. In other words, alignment problem consists in finding the

Received by the editors: July 26, 2010.

2000 *Mathematics Subject Classification.* 68T05, 91E45.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Concept learning*.

Key words and phrases. Concept alignment, Machine Learning, Binary Classification, Support Vector Machine.

correspondences between the definitions of different dictionaries (standards) that refer to the same concept. This alignment of definitions, which is one of the goals of ASICOM project[1] as well, has certainly to improve the fusion between the business models of different companies.

In our case, the concepts we are trying to align are belonging to the custom area and they are defined in two standards: CCL08A (Centre for Trade Facilitation and Electronic Business standard or shorter CCL) and Customs WCO (World Customs Organization or shorter WCO). In order to automatically perform this alignment, several definitions are considered from the two dictionaries. Pairs of two definitions (that can define the same concept - and in this case we deal with two aligned definitions - or different concepts - unaligned definitions in this case) are formed. Thus, the alignment problem could be considered as a binary classification problem: the inputs of the classifier are the pairs (of two definitions) and the outputs are the labels "aligned" or "unaligned" corresponding to each pair.

As we have mentioned, aligning two definitions means actually to solve a binary classification problem. Several couples of definitions, which could be aligned or unaligned, are required, so that the classifier could learn to discriminate correctly such relationships. In order to perform this alignment as a classification approach, all the possible couples of definitions are considered from the mentioned dictionaries (CCL and WCO). In this way, if a dictionary contains n_1 definitions and the other dictionary contains n_2 definitions, then we will be obtained $n_1 * n_2$ couples of definitions (some of them are aligned couples, while others are unaligned couples). Taking into account that we deal with a classification problem, a Machine Learning algorithm could be used.

For our task of solving the terminology alignment problem two Machine Learning based algorithms were chosen: a k Nearest Neighbour (kNN) algorithm [2] and a Support Vector Machine (SVM) [22] that work by using a particular representation based on the similarities between two definitions. Even if the kNN is a simple and fast algorithm, it requires to establish a threshold value and to specify the alignment cardinality. Therefore, we decided to try an SVM-based approach also, since it can learn the optimal value of the threshold and is able to produce any alignment type. Several performance measures, borrowed from the information retrieval domain, are used in order to evaluate the quality of the automatic produced alignments: the accuracy, the precision, the recall and the F-measure of alignments.

The remaining of the paper is structured as follows: Section 2 gives a short review of different alignment models, Sections 3 and 4 presents our solution to

[1]ASICOM – Architecture de Système d'information Interopérable pour les industries du Commerce

the alignment problem and several numerical results, while Section 5 concludes the results.

2. Related work

To our knowledge, the alignment problem has been intensively studied in order to achieve an automatic translation of terminologies, providing us several alignment models. The alignment problem has to be considered from two important points of view:

- the structures that must be aligned and
- the manner in which this alignment is actually performed (the matching algoritm).

2.1. **The structures that must be aligned.** Regarding the structure of the alignment, two important levels of a dictionary could be identified: the sentence level and the ontology level.

2.1.1. *The sentence level.* The sentence level refers to the bag of words of that dictionary, but it is actually a special bag of words where not only the frequency of a word is important, but also other linguistic information about these words. The richness of human language allows people to express the same idea in many different ways; they may use different words of the same language to refer to the same entity or employ different phrases to describe the same concept. Furthermore, the same idea could be expressed in different languages. Sentence-aligned bilingual corpora are a crucial resource for training statistical machine translation systems. In this context, we discuss about multilingual alignment. Therefore, the problem of sentence alignment for monolingual corpora is a phenomenon distinct from alignment in parallel, but multilingual corpora.

Monolingual alignment – Sentence-aligned bilingual corpora are a crucial resource for training statistical machine translation systems. Several authors have suggested that large-scale aligned monolingual corpora could by similarly used to improve the performance of monolingual text-to-text rewriting systems, for tasks including summarization [15, 16] and paraphrasing [1, 20]. Most of the work in monolingual corpus alignment is in the context of summarization. In a single document summarization alignment between full documents and summaries written by human is used to learn rules from text compression. Marcu [16] computes sentence similarity using a cosine-based metric. Jing [15] identifies phrases that were cut and pasted together using a Hidden Markov Model with features incorporated word identity and positioning within sentences, by providing an alignment of a document and its summary. In the context of multi document summarization, SimFinder [13]

identifies sentences that convey similar information across input documents to select the summary content.

Multilingual alignment – To our knowledge, the problem of aligning sentences from parallel corpora has been intensively studied for automated translation. While much of the research has focused on the unsupervised models [3, 5, 11], a number of supervised discriminatory approaches have been recently proposed for automatic alignment [4, 18, 21]. Related to the use of linguistic information more recent work [19] shows the benefit of combining multilevel linguistic representations (enriching query terms with their morphological variants). By coupling Natural Language Processing and Information Retrieval (IR) the language is enriched by combining several levels of linguistic information through morphological (lemma, stem), syntactic (bigrams, trigrams) and semantic (terms and their morphological and/or semantic variants) analyses. Moreover, data fusion has been exhaustively investigated in the literature, especially in the framework of IR [8, 19]. The difficulty is to find a way to combine results of multiple searches conducted in parallel on a common data set for a given query in order to obtain higher performances than each individual search.

2.1.2. *The ontology level.* The ontology level is actually a generalisation of the first level that take into account for a given dictionary not only the words and their order in a sentence (when the words are in fact considered isolated elements), but also the relationships (syntactic, semantic or other relationship types) establish among the words/concepts. In other words, at this level a concept (or its definition) could be represented as a bag of concepts (by concept being understood the corresponding word and its relationships with other words/concepts).

At present, there exists a line of semi-automated schema matching and ontology integration systems, see for instance [12, 17]. Most of them implement syntactic matching. The idea of generic (syntactic) matching was first proposed by Phil Bernstein and implemented in the Cupid system [17], but COMA [12] is a generic schema-matching tool, which seems to be a more flexible architecture. COMA provides an extensible library of matching algorithms; a framework for combining obtained results, and a platform for evaluating of the effectiveness of different matchers.

2.2. **Matching algorithms.** We distinguish between matching algorithms at element-level and structure-level.

The element-level matching techniques focus on the entities of a dictionary (words or concepts).They compute matching elements by analyzing entities in isolation, ignoring their relations with other entities: string-based techniques (normalization techniques, substring or subsequence techniques,

path comparison), language-based techniques (based on NLP techniques using morphological properties of the input words) – these methods may be either intrinsic (using the internal linguistic properties of the instances, such as morphological and syntactic properties: tokenization, elimination or filtration, lemmatization, stemming, weighting) or extrinsic (requiring the use of external resources, e.g. lexicon-based and multilingual methods: synonymy or semantic similarity).

The structure-level techniques focus on the structure of the ontology. It computes mapping elements by analyzing how entities appear together in the structure corresponding to a dictionary: graph-based techniques (graph algorithms which consider the input as labelled graphs) and taxonomy-based techniques (are also graph algorithms which consider only the specialization relation; the intuition behind taxonomic techniques is that is-a links connect terms that are already similar, therefore their neighbours may be also somehow similar).

3. Alignment methods

In order to obtain an automatic alignment two methods were utilized: the k Nearest Neighbour algorithm [2] and the Support Vector Machine [22].

3.1. **k Nearest Neighbour.**

kNN is a method of classifying objects based on closest training examples in a feature space. kNN is a type of instance based learning, or lazy learning where the function is only approximated locally and all the computation is deferred until classification. The kNN algorithm is amongst the simplest of all machine learning algorithms and is also very fast, two important reasons to utilize it for the alignment task. An object is classified by a majority vote of its neighbours, with the object being assigned to its class most common amongst its k nearest neighbours; k is a positive integer, typical small. If $k = 1$, then the object is simply assigned to its closest neighbour. In binary classification problems it's helpful to choose k to be an odd number as this avoids tied votes.

In our case an object corresponds to a definition (or to its representation as bag of "special" words at different syntactic levels). The distance between two objects (definitions) is computed by using a similarity measure. The smallest distance (or the largest similarity measure) between two definitions (from all the possible combinations) will indicate that the two definitions are aligned. The main idea of the kNN classifier in this case is to search amongst the couples of two definitions those with similarity greater than a given threshold and to select from the found couples the first k couples with the largest similarity. Such algorithm is able to produce two types of alignments:

- one-to-one alignments when $k = 1$ or

- one-to-many alignments when $k > 1$.

Even if kNN's methodology seems to be very simple, the usage of this algorithm in order to reach our purpose has determined several important questions:

- Which is the optimal value for k?
- Which is the optimal value of similarity threshold?

In order to solve these problems an SVM algorithm instead of the kNN classifier was used, since SVM can learn the optimal value for the threshold and is able to produce any alignment type:

- one-to-one – a concept of a dictionary corresponds to one concept of the other dictionary (*equivalence* relation);
- one-to-many – a concept of a dictionary corresponds to many concepts of the other dictionary (*type of* relation);
- many-to-many – more concepts of a dictionary corresponds to many concepts of the other dictionary.

3.2. **Support Vector Machine.** SVMs are a set of related supervised learning methods used for classification and regression. Generally, classification is defined for the situation when there are more objects, each one belonging to one of several classes, and a classification task would be to assign the belonging class to a new given object. In the case of binary classification using SVM, being given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. An SVM model is a representation of the examples as points in space, mapped by a kernel so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. In our case it is a binary classification problem, if the SVM is given a pair of definitions it will decide if they are aligned or not aligned.

An SVM algorithm has two phases: a training phase and a test phase. In the training phase the SVM model is learned starting from labelled examples (in our case, couples of definitions) and the hyper parameters are optimized and in the test phase the unseen definitions couples are labelled as aligned or unaligned.

Therefore, each data set has been divided in two: a part for training and a part for testing. The training part has been dived again in a learning subset, used by the SVM algorithm in order to find the hyper plane that makes the class separation and a validation set, used in order to optimize the values

of the hyper parameters. The SVM model, which is learned in this manner, classifies (labels) the definitions couples from the test set.

4. Numerical experiments

4.1. **Data representation.** Our input data is represented by definitions stored in the two dictionaries (WCO Dictionary and CCL Dictionary). These dictionaries are provided by the team of ASICOM project in a form of 3 levels taxonomies: a concept is represented by the text of the definition, the path in the hierarchy of concepts and the father of the concept. In order to use a Machine Learning algorithm, the natural language of definitions has to be processed. First, each sentence was turned into a bag of words and then followed these steps: normalisation, filtering of the stop words and lemmatization. Furthermore, each couple of definitions is represented by a set of similarity measures.

4.1.1. *First set of similarities.* In order to perform automatic alignments using the kNN algorithm several similarities computed at different levels of the ontology were used :

- Name: Synonyms and TriGrams similarity applied on concept;
- NameType: Name and Data Type similarity, applied on the concept;
- Path: Name similarity, applied on path of the concept in the given ontology;
- Leaves: Name Type applied on leaves of the concept in the given ontology;
- Children: Name Type applied on children of the concept in the given ontology;
- Comment: TriGrams similarity, applied on comment (definition) tokens.

These six distances were computed using the Coma++ tool [10]. The similarity is represented by a weighted sum of these six distances whose weights are optimized on the training set using an evolutionary algorithm.

4.1.2. *Second set of similarities.* An important step in using the SVM is building the input vector. For each couple of definitions there is built a vector that contains five similarity measures:

- the Match coefficient [7] that counts the common element for two definitions,
- the Dice coefficient [9] that is defined as twice the number of common words, divided by the total number of words from two definitions,
- the Jaccard [14] coefficient, which is defined as the number of common words, divided by the total number of words from two definitions,

- the Overlap [6] coefficient, which is defined as the number of common words, divided by the minimum of the number of words from two definitions,
- the Cosine measure that is defined as cosine similarity of the product between term frequency of a term in a definition and the inverse definition frequency:

$$cosine(def_i, def_j) = \frac{def_i * def_j}{||def_i|| * ||def_j||}.$$

In addition, this vector contains also the lengths of the definitions and the label (aligned or not-aligned).

Since Match, Dice, Jaccard, Overlap similarities are based on reunion (the total number of words from two definitions) and intersection (the common words of two definitions) a special way of computing this reunion and intersection is actually used in the numerical experiments. This special computation is adapted to the bag representation (instead of set-based representation). Considering the following definitions and the corresponding bag of words:

- def_i = (a, b, b, c, c, d),
- def_j = (a, a, b, b, b, c, e).

In this case:

- $def_i \cup def_j$ = (a, b, b, b, c, c, d, e),
- $def_i \cap def_j$ = (a, b, b, c)

As shown in the example, a word may appear several times in a definition and from this reason when we calculate the cardinal of intersection of two definitions we take the minimum occurrence of a word in the two definitions and while on calculating reunion, the maximum occurrence.

4.2. **Construction of the database.** Taking into account that both Machine Learning's algorithms require a training set and a test set, in the absence of a human expert, a database of 180 aligned couples of definitions was semi automatic created using the Coma++ tool [10]. Regarding the unaligned couples were selected 395 couples. The selection procedure was based on a normal distribution of the average distance over the unaligned couples. The training set contains 290 couples of definitions (90 aligned and 200 unaligned) and the test set contains 285 couples of definitions (90 aligned and 195 unaligned). This database was created by the ASICOM's team.

4.2.1. *k Nearest Neighbour.* kNN receives as input the similarities between two definitions by using the weighted sum of six similarity measures and outputs the performed alignments. In order to select the aligned couples we identify:

- those with similarity > threshold (many-to-many alignments) and

- the best alignment from those with similarity > threshold (one-to-many alignments).

The threshold was fixed 0.2 (default value from Coma++). The similarity value is represented by the weighted sum of the 6 distances presented in the first set of similarities. The weights are optimized on the training set, then validated on the test set by applying kNN with k=1.

4.2.2. *Support Vector Machine.* SVM receives as input the similarities between two definitions and outputs the label 1 if they are aligned and 0 if they are not aligned. The training set is built without mixing the couples of definitions, so first in the training set are taken the couples of aligned definitions and then the couples of unaligned definitions.

4.3. **Weights and parameters' optimization.** As we have previously mentioned, in the first experiment using kNN algorithm, the similarity is computed as a weighted sum of 6 distances. The weights are optimized using an evolutionary approach based on F-measure of aligned couples in the following manner: we consider a chromosome which codes the weights associated to the six distances. Its fitness is computed basing on the training set. Each definition from the first dictionary was compared with all the definitions from the second dictionary and was chosen the first couple with the greatest similarity value. If this similarity is greater than the threshold value, the couple is labelled as aligned, otherwise unaligned. In this manner 290 labelled couples are obtained. The fitness of the chromosome is represented by the F-measure of the aligned couples. The aligned couples are the ones that were considered aligned from the comparison with the threshold value and were initially labelled as aligned by Coma++, too. The best weights are the ones indicated by the chromosome with the greatest fitness value (F-measure). The best weights are learned on the training set and tested on the test set.

In the second experiment the SVM algorithm with an RBF kernel was used:

$$K(x, y) = exp(-\sigma|x - y|^2). \tag{1}$$

Cross Validation was made on the training set with different combinations of C and σ, C in range $[10^{-2}, 10^3]$ and σ in range $[2^{-5}, 2^2]$ by using a grid search algorithm and then the values from the combination (C and σ) with the best performances were chosen.

4.4. **Performance measures.** In order to measure the quality of the alignments the following performance measures were used: accuracy, recall, precision and F-measure, which are calculated for the aligned class (1) and the accuracy for both classes (1 and 0).

- the accuracy of alignments represents the percent of correct alignments from the total number of alignments;
- the precision of alignments represents the percent of relevant alignments among the proposed alignments;
- the recall of alignments represents the percent of relevant alignments that are effectively retrieved;
- F-measure represents the weighted harmonic mean of precision and recall. Greater F-measure (the maximal value being 1) signifies a correct and complete alignment.

4.5. **Numerical results.** In Table 1 the performances of the kNN and SVM based aligners are presented. The SVM classifier was applied at 3 different taxonomy levels while the kNN one utilizes a combination of them.

TABLE 1. The performance of the kNN and SVM-based aligners, using different similarity measures.

	Accuracy	Precision	Recall	F-measure
kNN	95%	87%	97%	92%
SVM on Explanation	32%	32%	100%	48%
SVM on Explanation & Path	90%	87%	81%	84%
SVM on Explanation & Path & Father	91%	87%	84%	86%

As we can notice from the results, by using the path and father of a concept it is possible to improve the performance of the classification process. The correct alignments were well recognized, but there are several not aligned definitions labelled as aligned by the SVM leading to a low precision, caused by: the couple type problems from our database like definitions with strong syntactic similarity and with different meaning (e.g. *Code specifying the type of package of an item* and *A code specifying a type of transport means*), the general-particular problem (e.g. *Means and mode of transport used for the carriage of the goods at departure, coded* and *A code specifying a type of transport means*) or words that appear in many definitions, but having different meaning.

The results using the kNN classifier on the definitions from this corpus are better than the ones using SVM, but, on the other hand, the utilization of the SVM classifier instead of the kNN one is more appropriate for solving the definition alignment problem taking into account that it can learn the optimal value of the threshold and it is able to produce any alignment type.

5. Conclusions

In this paper we presented our models for the automatic alignment of two terminologies (in fact, two real custom standards). These terminologies were reduced to several definitions taken from two dictionaries (CCL and WCO). The alignment issue was considered as a classification problem and solved by using two Machine Learning algorithms: kNN and SVM.

Even if the numerical results indicate that the SVM algorithm reaches weaker performances than the kNN method (from the F-measure performance point of view), the SVM is more helpful in order to align the given terminologies because it does not require fixing the values of the parameters. Furthermore, the SVM-based approach allows providing any type of alignments (one-to-one, one-to-many, many-to-many), which are very useful in the real world.

Future work will be focused on experiments considering a multi class problem. In this case we will deal with more types of definition couples: couples aligned and couples unaligned (straight unaligned, medium unaligned and weak unaligned).

References

[1] Barzilay, R., and Elhadad, N. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing* (2003).

[2] Bremner, D., Demaine, E., Erickson, J., Iacono, J., Langerman, S., Morin, P., and Toussaint, G. Output-sensitive algorithms for computing nearest-neighbor decision boundaries. *Discrete and Computational Geometry 4*, 33 (2005), 593–604.

[3] Brown, P. F., Pietra, S. D., Pietra, V. J. D., and Mercer, R. L. The mathematic of statistical machine translation: Parameter estimation. *Computational Linguistics 19*, 2 (1994), 263–311.

[4] Ceausu, A., Stefanescu, D., and Tufis, D. Acquis communautaire sentence alignment using Support Vector Machines. In *Proceedings of the 5th LREC Conference* (2006), pp. 2134–2137.

[5] Chen, S. F. Aligning sentences in bilingual corpora using lexical information. In *Meeting of the Association for Computational Linguistics* (1993), ACL, pp. 9–16.

[6] Clemons, T. E., and Bradley, E. L. A nonparametric measure of the overlapping coefficient. *Comput. Stat. Data Anal. 34* (2000), 51–61.

[7] Cormen, T., Leiserson, C., and Rivest, R. *Introduction to Algorithms*. MIT Press, 1990.

[8] Croft, W. B. *Combining approaches to information retrieval*. Springer, 2000, ch. 1, pp. 1–36.

[9] Dice, L. Measures of the amount of ecologic association between species. *Ecology 26*, 3 (1945), 297–302.

[10] Do, H.-H., and Rahm, E. Coma++ (combination of schema matching approaches), 2010.

[11] Gale, W. A., and Church, K. W. A program for aligning sentences in bilingual corpora. In *ACL* (1991), MIT Press, pp. 177–184.
[12] Hai-Do, H., and Rahm, E. COMA - A system for flexible combination of schema matching approaches. In *VLDB* (2002), P. A. Bernstein, Ed., Morgan Kaufmann, pp. 610–621.
[13] Hatzivassiloglou, V., Klavans, J., and Eskin, E. Detecting text similarity over short passages: Exploring linguistic feature combination via machine learning. In *Proceedings of the joint SIGDAT Conference on EMNLP/VLC-1999* (1999).
[14] Jaccard, P. The distribution of the flora of the alpine zone. *New Phytologist 11* (1912), 37–50.
[15] Jing, H. Using hidden markov modeling to decompose human-written summaries. *Computational Linguistics 28*, 4 (2002), 527–543.
[16] Knight, K., and Marcu, D. Statistics-based summarization - step one: Sentence compression. In *AAAI/IAAI* (2000), AAAI Press / The MIT Press, pp. 703–710.
[17] Madhavan, J., Bernstein, P. A., and Rahm, E. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases(VLDB '01)* (2001), P. M. G. Apers et al., Ed., Morgan Kaufmann, pp. 49–58.
[18] Moore, R. Fast and accurate sentence alignment of bilingual corpora. In *AMTA '02* (2002), S. D. Richardson, Ed., Springer, pp. 135–144.
[19] Moreau, F., Claveau, V., and Sébillot, P. Automatic morphological query expansion using analogy-based machine learning. In *ECIR 2007* (2007), G.Amati, C. Carpineto, and G. Romano, Eds., vol. 4425 of *LNCS*, Springer, pp. 222–233.
[20] Quirk, C., and Menezes, A. Dependency treelet translation: the convergence of statistical and example-based machine-translation? *Machine Translation 20*, 1 (2006), 43–65.
[21] Taskar, B., Lacoste, S., and Klein, D. A discriminative matching approach to word alignment. In *HLT '05* (2005), Association for Computational Linguistics, pp. 73–80.
[22] Vapnik, V. *The Nature of Statistical Learning Theory*. Springer, 1995.

[1]LITIS, EA - 4108, INSA, Rouen, France,

[2]Computer Science Department, Babeş Bolyai University, Cluj Napoca, Romania

E-mail address: adela_sarbu25@yahoo.com, lauras@cs.ubbcluj.ro
E-mail address: arogozan@insa-rouen.fr, pecuchet@insa-rouen.fr