Relational Feature Engineering of Natural Language Processing

Hamidreza Kobdani
Stuttgart University, Germany
kobdani@ims.uni-stuttgart.de

Hinrich Schütze
Stuttgart University, Germany
hs999@ifnlp.org

Andre Burkovski
Stuttgart University, Germany
burkovski@vis.uni-stuttgart.de

Wiltrud Kessler
Stuttgart University, Germany
kesslewd@vismail.informatik.uni-stuttgart.de

Gunther Heidemann
Stuttgart University, Germany
heidemann@vis.uni-stuttgart.de

ABSTRACT

We present a new framework for feature engineering of natural language processing that is based on a relational data model of text. It includes fast and flexible methods for implementing and extracting new features and thereby reduces the effort of creating an NLP system for a particular task.

In an instantiation and evaluation of the framework for the problem of coreference resolution in multiple languages, we were able to obtain competitive results in a short implementation period. This demonstrates the potential power of our framework for feature engineering.

Categories and Subject Descriptors

I.2.7 [**ARTIFICIAL INTELLIGENCE**]: Natural Language Processing; H.2.8 [**Information Systems**]: DATABASE MANAGEMENT—*Database applications*

General Terms

Design, Languages, Theory

Keywords

Natural Language Processing, Feature Engineering, Coreference Resolution, Relational Data Model

1. INTRODUCTION

We present a new framework for feature engineering of natural language processing (NLP) that is based on representing unstructured text in a relational data model. It includes powerful and flexible methods for implementing and extracting new features and thereby reduces the time and effort needed for creating a natural language processing system.

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

We aim to generalize previous approaches that use the relational model for NLP to a framework that can represent more complex linguistic structures and supports more complex feature definitions. At its core is a novel systematic approach for defining features of the textual entities in NLP.

We are interested in solving NLP problems using statistical classifiers trained in a supervised setting. The design and realization of a system in this approach involves the following steps, which we illustrate below using the problem of coreference resolution (CR) – the process of finding discourse entities, so-called *markables*, referring to the same real-world entity or concept.

1. Define classification problem. In the case of CR: the markable pairs are classified as coreferent or disreferent.

2. Provide training set. In CR, this is often text in which coreferent markable chains have been annotated.

3. Define evaluation measure. For CR, there are different measures. e.g. MUC [13], B^3 [1] and CEAF [10].

4. Select learning method and classifier. (e.g., decision tree, multinomial Naive Bayes, tree-kernel SVM etc)

5. Develop feature representation for instances. This is usually done through many iterations, in each of which the current feature representation is evaluated and analyzed on a validation set and then modified. This step is often the most difficult and time consuming and at the same time critical for building a successful system. To a large extent, the accuracy of classification in NLP depends on the quality of the features.

6. Final test. Test of performance of complete system on a held-out test set

In this paper, we present a framework that is intended to support the process of designing and realizing a machine-learning based NLP system in a number of ways.

(i) The heart of our framework is the relational data model that provides a common structure in a uniform and well-understood format of all the data that are used at any stage during the process: the text corpus, preprocessing results, relations between entities, feature definitions, feature values, classification results etc. As a result, there is more coherence of the methods and algorithms; and implementation, maintenance, reuse of components and integration of external components are all facilitated.

(ii) The relational model allows us to use SQL as the formalism for defining and extracting features. SQL is fast and

powerful. It supports a rapid development cycle in which a feature can be defined, tested and modified in a few minutes. This greatly reduces the effort of creating an NLP system for a particular task.

(iii) Our framework also makes it easy for the developer to reduce the number of features in cases where this is desirable. Removing redundant features often improves classification accuracy. Smaller feature sets also can be more robust and cut down classification times. If the latter is the main motivation for removing features, the developer can rapidly remove those features that have the smallest effect on classification accuracy.

(iv) The framework's modular architecture provides a clean separation between data storage, feature engineering and machine learning algorithms. As a result, it is easy to integrate any additional machine learning algorithm into the system.

(v) Many algorithms from the knowledge discovery and data mining community are intended and implemented for relational databases. Since we are using a relational model for all our data, it is possible to use these methods for NLP with little design and implementation effort.

To demonstrate the utility of our framework in NLP feature engineering, we have presented an instantiation and evaluation of the framework for the problem of CR in multiple languages [8]. A system (SUCRE) built with the framework achieved competitive results in a short implementation period doing a full CR of named entities, pronouns, and full noun phrases on the data sets of the six languages from SemEval-2010 Task 1 Coreference Resolution in Multiple Languages .

This paper is organized as follows. The data model is conceptualized in Section 2. Section 3 describes our approach to feature engineering. The Framework Architecture is introduced in Section 4. Sections 5 and 6 present related work and conclusions.

2. DATA MODEL

Relational data modeling is a well-known method for structured data and is supported by a wide range of software and many tools. The main purpose of the relational model in our framework is the use of a language like SQL [3] for feature definition. Using SQL, which is originally based on relational algebra (an offshoot of first-order logic), as the feature definition language is an easy to understand and flexible method for extracting different features from the relational data model of the text corpus.

The relational data model is the basis for defining the underlying structures of the text. To the extent that features in NLP are based on values of attributes of textual entities and on relations between entities, the relational database model is a natural formalism for supporting the definition and extraction of features. Converting a text corpus to its equivalent relational data model in a preprocessing step is simple and efficient as we will show below.

The main relation (table)[1] of the data model is built of tokens. This relation has one tuple (row) per token indexed from the beginning of the corpus. This tuple contains the token itself and its attributes (e.g. part-of-speech and lemma). According to this main relation (which is enough to rebuild a text corpus) we define other relations based on the underlying structures and their attributes. These relations model text structures such as documents, paragraphs, sentences, noun phrases and named entities and are defined on top of the token relation. For example, each tuple of the noun phrase relation contains, in addition to its attribute values, the indexes of its first, last and head words. All these relations and the corresponding attributes are used to define some new attributes in order to finally arrive at a desired set of attributes that act as the input feature vector to the employed learning algorithm. The basic relations are defined below:

Token (Token-ID,Sentence-ID,Token,Attribute-Vector)

Sentence (Sentence-ID,Paragraph-ID,Attribute-Vector)

Paragraph (Paragraph-ID,Document-ID, Attribute-Vector)

Document (Document-ID,Attribute-Vector)

For each specific NLP task, new task specific relations are needed. For example for CR, we have defined a new relation Markable as follows:

Markable (Markable-ID,First-Token-ID,Last-Token-ID, Head-Token-ID,Attribute-Vector)

It is evident that SQL can be used to manage the text data. For example with the following SQL statement we can access the words of a markable X:

```
SELECT * from Token INNER JOIN Markable ON
 Markable-ID = X AND (Token-ID >= First-Token-ID
                 AND  Token-ID <= Last-Token-ID)
```

After defining the basic and task specific relations for a particular NLP task, we need four relations for learning and classification (concerning samples and their corresponding feature vectors) as follows:

1. TrainSamples (Sample-ID, {Relational Connections} , Class)

2. TrainSamplesFeatureVector (Sample-ID, Feature-Vector)

3. TestSamples (Sample-ID, {Relational Connections}, Estimated-Class)

4. TestSamplesFeatureVector (Sample-ID, Feature-Vector)

In **TrainSamples** and **TestSamples**, *Relational Connections* is the set of referential constraints (foreign keys) that must be designed for a particular NLP task.

For example, if the training samples are pairs of markables, then *Relational Connections* are two foreign keys, Markable-1-ID and Markable-2-ID, and the TrainSamples table would be as follows:

TrainSamples (Sample-ID, Markable-1-ID, Markable-2-ID, Class)

Feature-Vector in **TrainSamplesFeatureVector** and **TestSamplesFeatureVector** is a vector of attributes that is defined based on the relations of the model. Each element of this vector is defined as an SQL statement that calculates its value. An example feature for the above mentioned TrainSamples would be: *the head word of the markable is a proper noun.* The simplified formulation of this feature in SQL would be (sample Y refers to markable X via Markable-ID):

[1]We use the formal terms *relation*, *tuple* and *attribute* for *table*, *row* and *column*.

```
UPDATE TrainSamplesFeatureVector
SET Feature-Vector[1] =
 SELECT * from Token
  INNER JOIN Markable
  ON Markable-ID = X AND
  (Token-ID == Head-Token-ID AND
   Token.Attribute-Vector[1] == "NP")
WHERE TrainSamplesFeatureVector.Sample-ID = Y
```

In the above SQL example, the token attribute 1 (`Token.Attribute-Vector[1]`) is part of speech and `"NP"` means proper noun.

3. FEATURE ENGINEERING

SQL includes operators on relations that we use to define features, including arithmetic, concatenation, comparison, logical, set and user-defined operators. Each feature definition is an attribute (column) of a relation (the feature vector table). In our setup, the feature extraction process consists of the calculation of features and their storage in the feature vector table.

The key capabilities of the relational model that we exploit are (i) all all data is available in a uniform relational model and (ii) SQL operators provide a uniform and powerful way of manipulating the data, in particular, for defining features. As a result, the developer can easily and quickly define features and thereby explore the feature space.

In addition to SQL operators, we can define our own functions such as the edit distance calculator or the feature binarizer. And in addition to features that are directly defined and computed on the relations, it is also possible to import externally generated features. For this purpose, it is enough to relate an externally generated feature value to its corresponding textual entity. This approach to feature engineering is suitable when a complex preprocessing pipeline provides complex features for the NLP task.

For example, a semantic similarity measure that is calculated by use of a search engine can be imported as an external feature and be combined with any other available feature in the system.

After its initial definition, the features are engineered with the goal of creating a feature set that will result in good performance. We view feature engineering as a process that consists of two parts: (1) Feature definition: which can be done by a feature definition language (SQL or pseudo-language) (2) Feature selection: we can consider two models of feature selection [9]: the *filter model* and the *wrapper model*. In the filter model, there is a set of features, and the final feature set is the subset that gives the best learning result. In the wrapper approach, a method for semiautomatic exploration of the feature space is used for finding better features. Our method of feature engineering can be categorized as a wrapper model, where we are able to search the feature space by defining new features or combining them to find the best possible feature set according to the result of training and evaluation data sets.

Feature definition and selection in our setup is an iterative process that consists of the following steps.

1. Definition of new features (new definition from scratch or definition by combination of existing features)
2. Evaluation of the features using the figure of merit (e.g. expected gain ratio [12])
3. Inclusion of promising features in the current feature set, removal of non-performing features
4. Data analysis to determine which phenomena are not captured by the current feature set

4. FRAMEWORK ARCHITECTURE

The architecture of the system has two main parts:

1. **Data Modeling**
   In a typical NLP system this part is known as preprocessing. In this part the text corpus is processed and converted to a relational data model. These are the main functionalities in this stage:
   (a) Preliminary text conversion (e.g. tokenization)
   (b) Extracting atomic attributes of tokens and basic structures (sentences, paragraphs and documents)
   (c) Task specific structure detection (e.g. markables)
   (d) Extracting atomic attributes of task specific structures (e.g. syntactical role of noun phrases)
2. **Classification**
   After relational modeling of the text corpus, the classification can be done. Its components are:
   (a) Instance generation: it can be interpreted as the sampling step in statistical learning.
   (b) Feature definition and extraction: here the feature definition and engineering methods which we described in section 2 and 3 are applied.
   (c) Learning (training): Any machine learning method that accepts the feature representation defined in the previous step can be used.
   (d) Classification (test): The classifier learned by the machine learning method is applied.

SUCRE [8] is an instantiation of this framework for the problem of CR in multiple languages. Table 1 presents some tables of the relational data model which we used in SUCRE.

As we showed before, the features can be directly defined in SQL, but this would make the system cumbersome to use for the average NLP developer. Therefore, we also define a pseudo-language which is syntactically more convenient. In particular, it contains keywords for the entities we need to refer to. These keywords are presented in the *Abbreviation* column of Table 1. Roughly speaking, the pseudo-language embeds all the needed inner joins to provide a fluent access to the attributes of different relations. The following are examples of two features in pseudo-language:

```
LFVT.f1 =
 (LT.mid1.tidf.stc == LT.mid2.tidf.stc)  &&
 (LT.mid2.tidh.a6 == reflexive)
```

It means: feature 1 is set to 1 if two markables are in the same sentence and the type of the second markable head word is reflexive (a6 is pronoun type in SUCRE); else 0.

```
LFVT.f2 =
 (LT.mid1.tidh.str == LT.mid2.tidh.str) &&
 (LT.mid1.tidh.a1 == N) &&
 (LT.mid2.tidh.a1 == N)
```

Column	Characteristic	Abbreviation
Token Table		TT
Token-ID	Primary Key	tid
Sentence-ID	Foreign Key	stc
Word-String	Attribute	str
Attribute-1	Attribute	a1
...	Attribute	...
Attribute-N	Attribute	an
Markable Table		MT
Markable-ID	Primary Key	mid
First-Token-ID	Foreign Key	tidf
Last-Token-ID	Foreign Key	tidl
Head-Token-ID	Foreign Key	tidh
Link Table (Train/Test Samples)		LT
Link-ID	Primary Key	lid
Markable-1-ID	Foreign Key	mid1
Markable-2-ID	Foreign Key	mid2
Class	Attribute	cls
Link Feature Vector Table		LFVT
Link-ID	Foreign Key	lid
Feature-1	Attribute	f1
...	Attribute	...
Feature-N	Attribute	fn

Table 1: Relational Data Model of Text Corpus for CR

It means: feature 2 is set to 1 if the head words of the two markables exactly match and are nouns (a1 is part of speech in SUCRE); else 0.

5. RELATED WORK

Early work on the application of relational models to facilitate text processing includes Crawford's approach [4], where it was shown that SEQUEL (the first version of SQL) can be used for keyword search. Searching in relational textual databases and user-defined operators to manage text were discussed by Lynch [11]. Grossman introduced a database design for document retrieval within the relational model [5]. He also uses relational databases for keyword-based information retrieval [6]. We are not aware of other work that uses the relational model for fast, interactive feature engineering in NLP.

The well-known WEKA [7] data mining software has a SQL viewer user interface that allows users to extract data from a database. But WEKA does not support the rapid iterative feature engineering that is the main purpose of our framework; for example, it does not offer an interactive mechanism to define features that can seamlessly access relational data. However, our framework could be easily integrated into WEKA for feature definition and extraction for NLP tasks. An example of such an adaptation for geographic data processing is proposed in [2] where an interoperable module is presented that uses a Geographic Database Management System to model distance and topological spatial relationships.

6. CONCLUSION

We have presented a new framework for feature engineering of NLP that is based on representing unstructured text in a relational database model. It includes powerful and flexible methods for implementing and extracting new features. It allows systematic and fast search of the space of features and thereby reduces the time and effort needed for creating an NLP system. The relational database model at the heart of our framework makes it possible to use SQL as the feature definition language.

The results of an instantiation of our framework for the problem of CR in multiple languages show that our approach to feature engineering supports rapid and high-quality development of a machine-learning based NLP system.

7. REFERENCES

[1] A. Bagga and B. Baldwin. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566, 1998.

[2] V. Bogorny, A. T. Palma, P. Engel, and L. O. Alvares. Weka-gdpm: Integrating classical data mining toolkit to geographic information systems. In *WAAMD*, pages 9–16. SBC, 2006.

[3] T. Connolly and C. E. Begg. *Database Systems: A Practical Approach to Design, Implementation and Management 2nd Ed.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[4] R. G. Crawford and I. A. Macleod. A relational approach to modular information retrieval systems design. In *Proceedings of the American Society for Information Systems Annual Meeting*, volume 15, 1978.

[5] D. Grossman. Using the relational model and part-of-speech tagging to implement text relevance. In *ACM CIKM*, 1992.

[6] D. Grossman, O. Frieder, D. O. Holmes, and D. C. Roberts. Integrating structured data and text: A relational approach. *Journal of the American Society of Information Science*, 48, 1997.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update, 2009.

[8] H. Kobdani and H. Schütze. Sucre: A modular system for coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 92–95, Uppsala, Sweden, July 2010. ACL.

[9] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[10] X. Luo. On coreference resolution performance metrics. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 25–32, Morristown, NJ, USA, 2005. ACL.

[11] C. A. Lynch and M. Stonebraker. Extended user-defined indexing with application to textual databases. In *Fourteenth International Conference on Very Large Data Bases, August 29 - September 1, 1988, Los Angeles, California, USA, Proceedings*, pages 306–317, 1988.

[12] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[13] M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. A model-theoretic coreference scoring scheme. In *MUC6 '95: Proceedings of the 6th conference on Message understanding*, pages 45–52, Morristown, NJ, USA, 1995. ACL.