Recognizing Documents versus Meta-Documents by Tree Kernel Learning

Boris Galitsky and Nina Lebedeva
Knowledge-Trail Inc. San Jose CA USA
{bgalitsky@hotmail.com, nina.lebedevakt@gmail.com}

Abstract

The problem of classifying text with respect to metalanguage and language object patterns is formulated and its application areas are proposed. Examples of metalanguage patterns in text are foreign language grammar lessons and tutorials on how to write engineering documents. The method targets the text classification tasks where keyword statistics is insufficient do distinguish between such abstract classes of text as metalanguage and object-level. To do that, we extend the parse tree kernel method from the level of individual sentences towards the level of paragraphs. We build a set of extended trees for a paragraph of text merging individual parse trees for sentences. We evaluate our approach in the domain of the design documents, differentiating them from meta-documents such as instructions on how to write design documents.

Introduction

Usually, in the majority of text classification problems, keywords statistics is sufficient to determine a class. Keywords are sufficient information to determine a topic of a text or document, such as software vs hardware, or pop rock vs punk. However, there are classification problems where distinct classes share the same keywords, and document phrasing, style and other kinds of text structure information needs to be taken into account. To perform text classification in such domain, discourse information such as anaphora and rhetoric structure needs to be taken into account.

We are interested in classifying a text with respect to being metalanguage or language object. If a text tells us how to do things, or how something has been done, we relate this text to a language object. If a text is saying how to write a document which explains how to do things, we related it to metalanguage. Metalanguage is a language or symbolic system used to discuss, describe, or analyze another language or symbolic system (Štuikys and Damaševicius 2013). Logic programs can be recognized as meta-programs or object-level programs easily (Galitsky 2003). However in a natural language text one needs some implicit cues at the syntactic level to recognize metalanguage patterns.

Obviously, using just keyword information would be insufficient to differentiate between texts in metalanguage and language-object. A presence of verbs for communicative actions and mental states may help to identify metalanguage patterns, but is still insufficient: *I know the height of the tallest skyscraper* vs *I know what she thinks about the skyscrapers*, the latter containing the meta-predicate ranging over a set of expressions for thoughts about skyscrapers. Use of parse trees would give us specific phrases in use by texts in metalanguage, but still will not be sufficient for systematic exploration of metalanguage-related linguistic features. It is hard to identify them unless one can analyze the discourse structure, including anaphora, rhetoric relations, and interaction scenarios by means of communicative language. Furthermore, to systematically learn these discourse features associated with metalanguage, we need a unified approach to classify graph structures at the level of paragraphs (Galitsky 2013).

The design of syntactic features for automated learning of syntactic structures for classification is still an art nowadays. One of the solutions to systematically treat these syntactic features is the approach of tree kernels built over syntactic parse trees. Convolution tree kernel (Collins and Duffy, 2002) defines a feature space consisting of all subtree types of parse trees and counts the number of common subtrees as the syntactic similarity between two parse trees. They have found a number of applications in a number of NLP tasks, including search (Moschitti, 2006), syntactic parsing re-ranking, relation extraction Zelenko et al 2003), named entity recognition (Cumby & Roth 2003) and Semantic Role Labeling relation extraction (Zhang et al., 2006), pronoun resolution (Yang et al., 2006), question classification and machine translation (Zhang and Li, 2009).

An approach to build a kernel based on more than a single parse tree has been proposed (Galitsky et al 2013, Galitsky 2014) with the focus on search. To perform a classification task based on additional discourse features, we form a single tree from a tree forest for a sequence of sentences in a paragraph of text. Currently, kernel methods tackle individual sentences. For example, in question answering, when a query is a single sentence and an answer is a single sentence, these methods work fairly well. However, in learning settings where texts include multiple sentences, one needs to go beyond the sentence boundaries and represent structures which include paragraph-level discourse information.

A number of NLP tasks such as classification require computing of semantic features over paragraphs of text containing multiple sentences. Doing it in at the level of individual sentences and then summing up the score for sentences will not always work. In the complex classification tasks where classes are defined in an abstract way, the difference between them may lay at the paragraph level and not at the level of individual sentences. In the case where classes are defined not via topics but instead via writing style, discourse structure signals become essential. Moreover, some information about entities can be distributed across sentences, and classification approach needs to be independent of this distribution. We will demonstrate the contribution of paragraph-level approach vs the sentence level in our evaluation. We will apply the extended tree kernel learning to classify engineering documents vs the rules on how to write these documents.

We define design document as a document which contains a thorough and well-structured description of how to build a particular engineering system. In this respect a design doc according to our model follows the reproducibility criteria of a patent or research publication; however format is different from them. What we exclude is a document which contains meta-level information relatively to the design of engineering system, such as how to write design docs manuals, standards design docs should adhere to, tutorials on how to improve design documents, and others.

We need to differentiate design documents from the classes of documents which can be viewed as ones containing meta-language, whereas the genuine design documents consists of the language-object patterns and should not include metalanguage ones. Meta-documents include instructions and recommendations on how to write design documents, project requirement document, requirement analysis, operational requirements, construction documentation, project planning and binning, technical services review, guidelines, manuals and others.

Extending tree kernels beyond sentences

Why can sentence-level tree kernels be insufficient for classification? The signals of meta-language in use are distributed through multiple sentences and are frequently invisible at the sentence level. We combine/merge parse trees for individual sentences to make sure we cover the phrases of interest. For example, for the following text:
This document describes the design of back end processor. Its requirements are enumerated below. From the first sentence, it looks like we got the design doc. To process the second sentence, we need to disambiguate the preposition 'its'. As a result, we conclude from the second sentence that it is a *requirements doc*, not a *design doc*.

The structure of NL which can be potentially valuable for classification can be characterized by rhetoric relations that hold between the parts of a text. These relations, such as explanations or contrast, are important for text understanding in general since they contain information on how these parts of text are related to each other to form a coherent discourse. Naturally, we expect the structure of discourse for metalanguage patterns to be different to that of language-object patterns. Discourse analysis explores how meanings can be built up in a communicative process, which is different for a metalanguage and a language-object. Each part of a text has a specific role in conveying the overall message of a given text.

We introduce a domain where a pair-wise comparison of sentences is insufficient to properly learn certain semantic features of texts. This is due to the variability of ways information can be communicated in multiple sentences, and variations in possible discourse structures of text which needs to be taken into account.

We consider an example of text classification problem, where short portions of text belong to two classes:

- Tax liability of a landlord renting office to a business.
- Tax liability of a business owner renting an office from landlord (shown in greyed area below).

I rent an office space. This office is for my business. I can deduct office rental expense from my business profit to calculate net income.

To run my business, I have to rent an office. The net business profit is calculated as follows. Rental expense needs to be subtracted from revenue.

To store goods for my retail business I rent some space. When I calculate the net income, I take revenue and subtract business expenses such as office rent.

I rent out a first floor unit of my house to a travel business. I need to add the rental income to my profit. However, when I repair my house, I can deduct the repair expense from my rental income.

I receive rental income from my office. I have to claim it as a profit in my tax forms. I need to add my rental income to my profits, but subtract rental expenses such as repair from it.

I advertised my property as a business rental. Advertisement and repair expenses can be subtracted from the rental income. Remaining rental income needs to be added to my profit and to be reported as taxable profit.

Note that keyword-based analysis does not help to separate the first three paragraph and the second three paragraphs. They all share the same keywords *rental / office / income / profit / add / subtract.* Phrase-based analysis does not help either, since both sets of paragraphs share similar phrases.

Pair-wise sentence comparison does not solve the problem.

Anaphora resolution is helpful but insufficient. All these sentences include 'I' and its mention, but other links between words or phrases in different sentences need to be used.

Rhetoric structures need to come into play to provide additional links between sentences. The structure to distinguish between

- *renting for yourself and deducting from total income* and
- *renting to someone and adding to income* embraces multiple sentences.

The second clause about *adding/subtracting incomes* is linked by means of the rhetoric relation of *elaboration* with the first clause for *landlord/tenant*. This rhetoric relation may link discourse units within a sentence, between consecutive sentences and even between first and third sentence in a paragraph. Other rhetoric relations can play similar role for forming essential links for text classification.

Which representations for these paragraphs of text would produce such common sub-structure between the structures of these paragraphs? We believe that extended trees, which include the first, second, and third sentence for each paragraph together can serve as a structure to differentiate the two above classes.

The dependency parse trees for the first text in our set and its coreferences are shown below:

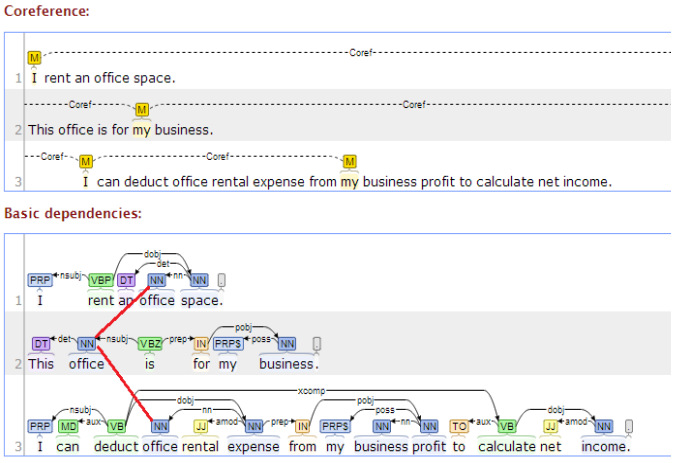
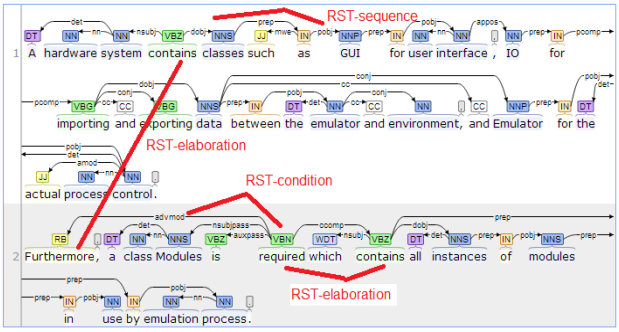


Fig. 1: Anaphora (on the top) and two rhetoric relations of elaboration (on the bottom) which link the sentences and help us to form a class-determining structure

There are multiple ways the nodes from parse trees of different sentences can be connected: we choose the rhetoric relation of elaboration which links the same entity office and helps us to form the structure *rent-office-space – for-my-business – deduct-rental-expense* which is the base for our classification.

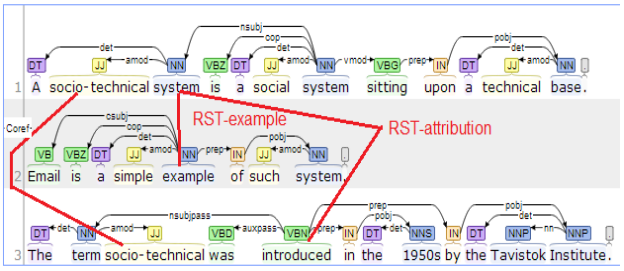
We proceed to the second example. Given a positive sequence and its parse trees linked by RST relations:

A hardware system contains classes such as GUI for user interface, IO for importing and exporting data between the emulator and environment, and Emulator for the actual process control. Furthermore, a class Modules is required which contains all instances of modules in use by emulation process.



and a negative sequence and its linked parse trees:

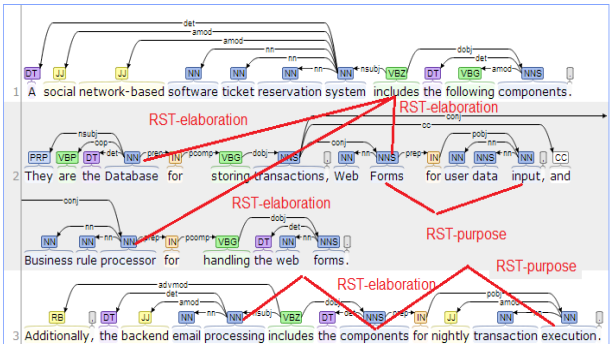
A socio-technical system is a social system sitting upon a technical base. Email is a simple example of such system. The term socio-technical was introduced in the 1950s by the Tavistok Institute.



We want to classify the paragraph

A social network-based software ticket reservation system includes the following components. They are the Database for storing transactions, Web Forms for user data input, and Business rule processor for handling the web forms. Additionally, the backend email processing includes the components for nightly transaction execution.

One can see that it follows the rhetoric structure of the top (positive) training set element, although it shares more common keywords with the bottom (negative) element. Hence we classify it as a design doc text, since it describes the system rather than introduces a terms (as the negative element does).



Forming extended parse trees

For every inter-sentence arc which connects two parse trees, we derive the extension of these trees. There is a pair of such extensions (Fig. 2):

1) Starting from the first tree, we navigate it till we reach a connecting inter-sentence arc, then we use this arc to 'jump' to the second tree, which we continue to navigate towards the level of leaves;
2) Starting from the second tree, we navigate it till we reach a connecting inter-sentence arc, then we use this arc to 'jump' to the first tree, which we continue to navigate towards the level of leaves.

In this approach, for a given parse tree, we will obtain a set of its extension for each inter-sentence arc it connects this sentence to other sentences. Hence the elements of the kernel (sub-trees) will be computed for multiple extensions, instead of just a single tree (as happens under conventional tree kernel). The computational problem here is that we need to find common sub-trees for a much higher number of trees than the number of sentences in text, however by subsumption (sub-tree relation) the number of common sub-trees will be substantially reduced.

If we have two parse trees P_1 and P_2 for two sentences in a paragraph, and a relation R_{12}: $P_{1i} \rightarrow P_{2j}$ between the nodes P_{1i} and P_{2j}, we form the pair of extended trees $P_1 * P_2$:

$\ldots, P_{1i-2}, P_{1i-1}, P_{1i}, P_{2j}, P_{2j+1}, P_{2j+2}, \ldots$

$\ldots, P_{2j-2}, P_{2j-1}, P_{2j}, P_{1i}, P_{1i+1}, P_{2i+2}, \ldots,$

which would form the feature set for tree kernel learning in addition to the original trees P_1 and P_2.

The algorithm for building an extended tree for a set of parse trees T is as follows:

Input:
1) Set of parse trees T.
2) Set of relations R, which includes relations R_{ijk} between the nodes of T_i and T_j: $T_i \in T$, $T_j \in T$, $R_{ijk} \in R$. We use index k to range over multiple relations between the nodes of parse tree for a pair of sentences.
Output: the exhaustive set of extended trees E.

Set $E = \varnothing$;
For each tree i=1:|T|
  For each relation R_{ijk}, k= 1: |R|
    Obtain T_j
    Form the pair of extended trees $T_i * T_j$;
    Verify that each of the extended trees do not have a super-tree in E
    If verified, add to E;
Return E.

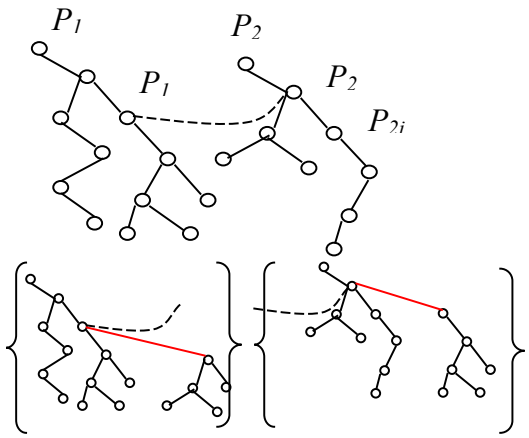


Fig. 2: An arc which connects two parse trees for two sentences in a text (on the top) and the derived set of extended trees (on the bottom).

Notice that the resultant trees are not the proper parse trees for a sentence, but nevertheless form an adequate feature space for tree kernel learning.

To obtain the inter-sentence links, we employed coreferences from Stanford NLP (Recasens et al 2013, Lee et al 2013). Rhetoric relation extractor based on our rule-based approach to finding relations between elementary discourse units (Galitsky et al 2013). We combined manual rules with automatically learned rules derived from the available discourse corpus by means of syntactic generalization.

Kernel methods are a large class of learning algorithms based on inner product vector spaces. Support vector machines (SVMs) are mostly well-known algorithms. Convolution kernels compute the common sub-trees between two trees T_1 and T_2. Convolution kernel does not have to compute the whole space of tree fragments. Let the

set

$$\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$$

be the set of sub-trees of an extended parse tree, and $\chi_i(n)$ be an indicator function which is equal to 1 if the subtree t_i is rooted at a node n, and is equal to 0 otherwise. A tree kernel function over trees T_1 and T_2 is

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2),$$

where N_{T1} and N_{T2} are the sets of T_1 's and T_2 's nodes, respectively and

$$\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{T}|} \chi_i(n_1)\chi_i(n_2).$$

It calculates the number of common fragments with the roots in n_1 and n_2 nodes.

Evaluation

For design documents, we built a web mining utility which searched for public design documents on the web in a number of engineering and science domains. We formed a set of 1200 documents, it turned out we had 90% of non-design engineering documents of the classes we want to exclude (meta-documents) and 10% of genuine design documents. We split the data into 5 sub-sets for training/evaluation portions. For the design documents, evaluation results were assessed by quality assurance personnel.

TF*IDF Nearest Neighbor approach finds a document in the training set which is the closest to the given one being recognized. Nearest Neighbor feature is implemented via the search in inverse index of Lucene (Croft et al 2009) where the search result score is computed based on TF*IDF model (Salton and Buckley 1988). The query is formed from each sentence of the documents being classified as a disjunctive query including all words except stop-words. The resultant classes along with their TF*IDF scores are weighted and aggregated on the basis of a majority vote algorithm such as (Moore and Boyer 1991).

A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions. This classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple. Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood. WEKA classifier is based on (John and Langley 1995).

We report the standard deviation of F-measure over five folds achieved by different methods in Table 1. Each row shows the results of a particular classification method. Keyword statistic-based methods, including Nearest-Neighbor classification and Naïve Bayes, produced rather poor results. Conversely, a manual rule-based system produces a very high accuracy result, especially when manually formed rules go beyond the keywords/phrases and take into account part-of-speech information.

An increase in accuracy by a few percent is achieved in design documents by using human comprehension of which expressions are used in metalanguage and style in meta-documents. These rules also included regular expressions relying on specific document formatting including a table of content and structure of sections. Performance of tree kernel based methods improves as the sources of linguistic properties expand. There is a few percent improvement by using RST relations compared with baseline tree kernel SVM which relies on parse trees only.

Method	Precision	Recall	F-measure	Standard deviation
Nearest neighbor classifier – tf*idf based	55.8	61.4	58.47	2.1
Naive Bayesian classifier (WEKA)	57.4	59.2	58.29	3.2
Tree kernel – regular parse trees	73.4	77.6	75.44	2.8
Tree kernel SVM – extended trees for anaphora	77.0	79.3	78.13	3.1
Tree kernel SVM – extended trees for RST	78.3	81.5	79.87	2.6
Tree kernel SVM – extended trees for both anaphora & RST	82.1	85.2	83.62	2.7

Table 1: Classifying text into metalanguage and language-object

Discussion and Conclusions

In our previous works it was observed how employing a richer set of linguistic information such as syntactic relations between words assists relevance tasks (Galitsky et al. 2012). To take advantage of semantic discourse information, we introduced parse thicket representation and proposed the way to compute similarity between texts based on generalization of parse thickets (Galitsky et al 2013). We build the framework for generalizing PTs as sets of phrases on one hand, and generalizing PTs as graphs via maximal common subgraphs, on the other hand (Galitsky 2013).

In this study we focused on how discourse information can help with a fairly abstract text classification tasks by means of statistical learning. We selected the domain where the only difference between classes lays in phrasing and discourse structures and demonstrated that both are learnable. We compared two sets of linguistic features:

- The baseline, parse trees for individual sentences,
- Parse trees and discourse information,

and demonstrated that the enriched set of features indeed improves the classification accuracy, having the learning framework fixed. We also demonstrated that the baseline text classification approaches perform rather poorly in the chosen classification domain. Kernel-based learning was unable to reach the performance of manually structure-based rules, and we hypothesize that a vast amount of discourse information is not employed in the proposed learning framework.

In our previous studies we considered the following sources of relations between words in sentences: coreferences, taxonomic relations such as sub-entity, partial case, predicate for subject etc., rhetoric structure relations, and speech acts (Galitsky 2013). We demonstrated that a number of NLP tasks including search relevance can be improved if search results are subject to confirmation by parse thicket generalization, when answers occur in multiple sentences. In this study we employed coreferences and rhetoric relation only to identify correlation with the occurrence of metalanguage in text. Although phrase-level analysis allows extraction of weak correlation with metalanguage in text, ascend to discourse structures makes detection of metalanguage more reliable. In our evaluation setting, using discourse improved the classification F-measure by 6 to 9%.

Applying tree kernels for a number of NLP applications such as search and classification of text, various authors including (Sun et al 2011) attempt to cover all linguistic features, such as keyword statistics. By building feature space which is intended to contain an exhaustive set of linguistic features, not only the structural ones represented by trees, these authors make it harder to estimate the contribution of the tree kernel approach compared to the competitive ones. In this study we applied tree kernel to such the domain where only structural information is relevant and keyword statistics is not. It turned out that the sentence-level structured information is insufficient for classification, so its extension beyond individual sentences was required.

The experimental environment, extended tree learning functionality, and the evaluation framework is available at http://code.google.com/p/relevance-based-on-parse-trees.

References

Štuikys, V., Damaševicius, R. 2013. Meta-Programming and Model-Driven Meta-Program Development Springer ISSN 1610-3947.

Galitsky, B. 2003. Natural language question answering system: Technique of semantic headers. Advanced Knowledge International, Australia.

Galitsky, B., Josep Lluis de la Rosa, Gábor Dobrocsi. 2012. *Inferring the semantic properties of sentences by mining syntactic parse trees*. Data & Knowledge Engineering. Volume 81-82, November, 21-45.

Galitsky, B. 2012. *Machine Learning of Syntactic Parse Trees for Search and Classification of Text*. 2012. Engineering Application of AI, http://dx.doi.org/10.1016/j.engappai.2012.09.017.

Galitsky, B., Dmitry Ilvovsky, Sergei O. Kuznetsov, Fedor Strok. 2013. *Improving Text Retrieval Efficiency with Pattern Structures on Parse Thickets*, in Workshop on Formal Concept Analysis meets Information Retrieval at ECIR 2013, Moscow, Russia.

Galitsky, B. 2014. Learning parse structure of paragraphs and its applications in search. Engineering Applications of Artificial Intelligence. 01/2014; 32:160–184.

Moschitti, A. 2006. *Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees*. In Proceedings of the 17th European Conference on Machine Learning, Berlin, Germany, 2006.

Sun, J., Min Zhang, Chew Lim Tan. *Tree Sequence Kernel for Natural Language*. AAAI-25, 2011.

Zhang, M.; Che, W.; Zhou, G.; Aw, A.; Tan, C.; Liu, T.; and Li, S. 2008. *Semantic role labeling using a grammar-driven convolution tree kernel*. IEEE transactions on audio, speech, and language processing 16(7):1315–1329.

Collins, M., and Duffy, N. 2002. *Convolution kernels for natural language. In Proceedings of NIPS*, 625–632.

Lee, H., Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu and Dan Jurafsky. 2013. *Deterministic coreference resolution based on entity-centric, precision-ranked rules*. Computational Linguistics 39(4).

Sun, J., Zhang, M.; and Tan, C. 2010. *Exploring syntactic structural features for sub-tree alignment using bilingual tree kernels*. In *Proceedings of ACL*, 306–315.

Recasens, M. Marie-Catherine de Marneffe, and Christopher Potts. The Life and Death of Discourse Entities: Identifying Singleton Mentions. In Proceedings of NAACL 2013.

Zelenko, D., Aone, C., Richardella, A. 2003. Kernel methods for relation extraction. JMLR.

Zhang, M. and Haizhou LI. 2009. Tree Kernel-based SVM with Structured Syntactic Knowledge for BTG-based Phrase Reordering. EMNLP-2009.