Using Syntactic and Semantic Structural Kernels for Classifying Definition Questions in Jeopardy!

Alessandro Moschitti[†] **Jennifer Chu-Carroll**[‡] **Siddharth Patwardhan**[‡]

James Fan[‡] **Giuseppe Riccardi**[†]

[†]Department of Information Engineering and Computer Science
University of Trento, 38123 Povo (TN), Italy
{moschitti,riccardi}@disi.unitn.it
[‡]IBM T.J. Watson Research Center P.O. Box 704, Yorktown Heights, NY 10598, U.S.A.
{jencc,siddharth,fanj}@us.ibm.com

Abstract

The last decade has seen many interesting applications of Question Answering (QA) technology. The Jeopardy! quiz show is certainly one of the most fascinating, from the viewpoints of both its broad domain and the complexity of its language. In this paper, we study kernel methods applied to syntactic/semantic structures for accurate classification of Jeopardy! definition questions. Our extensive empirical analysis shows that our classification models largely improve on classifiers based on word-language models. Such classifiers are also used in the state-of-the-art QA pipeline constituting Watson, the IBM Jeopardy! system. Our experiments measuring their impact on Watson show enhancements in QA accuracy and a consequent increase in the amount of money earned in game-based evaluation.

1 Introduction

Question Answering (QA) is an important research area of Information Retrieval applications, which requires the use of core NLP capabilities, such as syntactic and semantic processing for a more effective user experience. While the development of most existing QA systems are driven by organized evaluation efforts such as TREC (Voorhees and Dang, 2006), CLEF (Giampiccolo et al., 2007), and NTCIR (Sasaki et al., 2007), there exist efforts that leverage data from popular quiz shows, such as Who Wants to be a Millionaire (Clarke et al., 2001; Lam et al., 2003) and Jeopardy! (Ferrucci et al., 2010), to demonstrate the generality of the technology.

Jeopardy! is a popular quiz show in the US which has been on the air for 27 years. In each game, three contestants compete for the opportunity to answer 60 questions in 12 categories of 5 questions each. Jeopardy! questions cover an incredibly broad domain, from science, literature, history, to popular culture. We are drawn to Jeopardy! as a test bed for open-domain QA technology due to its broad domain, complex language, as well as the emphasis on accuracy, confidence, and speed during game play.

While the vast majority of Jeopardy! questions are factoid questions, we find several other types of questions in the Jeopardy! data, which can benefit from specialized processing in the QA system. The additional processing in these questions complements that of the factoid questions to achieve improved overall QA performance. Among the various types of questions handled by the system are *definition questions* shown in the examples below:

(1) GON TOMORROW*: It can be the basket below a hot-air balloon or a flat-bottomed boat used on a canal* (answer: gondola);

(2) I LOVE YOU, "MIN"*: Overbearing* (answer: domineering);

(3) INVEST*: From the Latin for "year", it's an investment or retirement fund that pays out yearly* (answer: an annuity)

where the upper case text indicates the Jeopardy! category for each question[1].

Several characteristics of this class of questions warrant special processing: first, the clue (question)

[1]A Jeopardy! category indicates a theme is common among its 5 questions.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 712–724,
Edinburgh, Scotland, UK, July 27–31, 2011.

often aligns well with dictionary entries, making dictionary resources potentially effective. Second, these clues often do not indicate an answer type, which is an important feature for identifying correct answers in factoid questions (in the examples above, only (3) provided an answer type, "fund"). Third, definition questions are typically shorter in length than the average factoid question. These differences, namely the shorter clue length and the lack of answer types, make the use of a specialized machine learning model potentially promising for improving the overall system accuracy. The first step for handling definitions is, of course, the automatic separation of definitions from other question types, which is not a simple task in the Jeopardy! domain. For instance, consider the following example which is a variation of (3) above:

(4) INVEST*: From the Latin for "year", an annuity is an investment or retirement fund that pays out this often* (answer: yearly)

Even though the clue is nearly identical to (3), the clue does *not* provide a definition for the answer *yearly*, although at first glance we may have been misled. The source of complexity is given by the fact that Jeopardy! clues are not phrased in interrogative form as questions typically are. This complicates the design of definition classifiers since we cannot directly use either typical structural patterns that characterize definition/description questions, or previous approaches, e.g. (Ahn et al., 2004; Kaisser and Webber, 2007; Blunsom et al., 2006). Given the complexity and the novelty of the task, we found it useful to exploit the kernel methods technology. This has shown state-of-the-art performance in Question Classification (QC), e.g. (Zhang and Lee, 2003; Suzuki et al., 2003; Moschitti et al., 2007) and it is very well suited for engineering feature representations for novel tasks.

In this paper, we apply SVMs and kernel methods to syntactic/semantic structures for modeling accurate classification of Jeopardy! definition questions. For this purpose, we use several levels of linguistic information: word and POS tag sequences, dependency, constituency and predicate argument structures and we combined them using state-of-the-art structural kernels, e.g. (Collins and Duffy, 2002; Shawe-Taylor and Cristianini, 2004; Moschitti, 2006). The extensive empirical analysis of several advanced models shows that our best model, which combines different kernels, improves the F1 of our baseline model by 67% relative, from 40.37 to 67.48. Surprisingly, with respect to previous findings on standard QC, e.g. (Zhang and Lee, 2003; Moschitti, 2006), the Syntactic Tree Kernel (Collins and Duffy, 2002) is not effective whereas the exploitation of partial tree patterns proves to be essential. This is due to the different nature of Jeopardy! questions, which are not expressed in the usual interrogative form.

To demonstrate the benefit of our question classifier, we integrated it into our Watson by coupling it with search and candidate generation against specialized dictionary resources. We show that in end-to-end evaluations, Watson with kernel-based definition classification and specialized definition question processing achieves statistically significant improvement compared to our baseline systems.

In the reminder of this paper, Section 2 describes Watson by focusing on the problem of definition question classification, Section 3 describes our models for such classifiers, Section 4 presents our experiments on QC, whereas Section 5 shows the final impact on Watson. Finally, Section 6 discusses related work and Section 7 derives the conclusions.

2 Watson: The IBM Jeopardy! System

This section gives a quick overview of Watson and the problem of classification of definition questions, which is the focus of this paper.

2.1 Overview

Watson is a massively parallel probabilistic evidence-based architecture for QA (Ferrucci et al., 2010). It consists of several major stages for underlying sub-tasks, including analysis of the question, retrieval of relevant content, scoring and ranking of candidate answers, as depicted in Figure 1. In the rest of this section, we provide an overview of Watson, focusing on the task of answering definitional questions.

Question Analysis: The first stage of the pipeline, it applies several analytic components to identify key characteristics of the question (such as answer

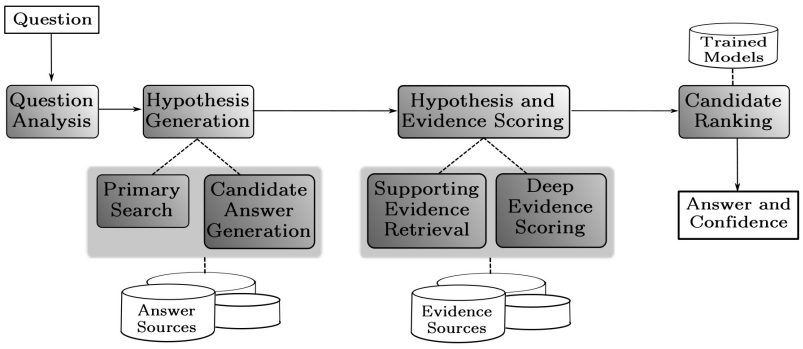

Figure 1: Overview of Watson

type, question classes, etc.) used by later stages of the Watson pipeline. Various general purpose NLP components, such as a parser and named entity detector, are combined with task-specific modules for this analysis.

The task-specific analytics include several QC components, which determine if the question belongs to one or more broad "question classes". These question classes can influence later stages of the Watson pipeline. For instance, a question detected as an *abbreviation* question can invoke specialized candidate generators to produce possible expansions of the abbreviated term in the clue. Similarly, the question classes can impact the methods for answer scoring and the machine learning models used for ranking candidate answers. The focus of this paper is on the *definition* class, which is described in the next section.

Hypothesis Generation: Following question analysis, the Watson pipeline searches its document collection for relevant documents and passages that are likely to contain the correct answer to the question. This stage of the pipeline generates search queries based on question analysis results, and obtains a ranked list of documents and passages most relevant to the search queries. A variety of candidate generation techniques are then applied to the retrieved results to produce a set of candidate answers.

Information obtained from question analysis can be used to influence the search and candidate generation processes. The question classes detected during question analysis can focus the search towards specific subsets of the corpus. Similarly, during candidate generation, strategies used to generate the set of candidate answers are selected based on the detected question classes.

Hypothesis and Evidence Scoring: A wide variety of answer scorers are then used to gather evidence supporting each candidate answer as the correct answer to the given question. The scorers include both context dependent as well as context independent scorers, relying on various structured and unstructured resources for their supporting evidence.

Candidate Ranking: Finally, machine learning models are used to weigh the gathered evidence and rank the candidate answers. The models generate a ranked list of answers each with an associated confidence. The system can also choose to refrain from answering a question if it has low confidence in all candidates. This stage of the pipeline employs several machine learning models specially trained to handle various types of questions. These models are trained using selected feature sets based on question classes and candidate answers are "routed" to the appropriate model according to the question classes detected during question analysis.

2.2 Answering Definition Questions

Among the many question classes that Watson identifies and leverages for special processing, of particular interest for this paper is the class we refer to as *definition* questions. These are questions whose clue texts contain one or more definitions of the correct answer. For instance, in example (3), the main clause in the question corresponds to a dictionary definition of the correct answer (*annuity*). Looking up this definition in dictionary resources could enable us to answer this question correctly and with high confidence. This suggests that special process-

ing of such definition questions could allow us to hone in on the correct answer through processes different from those used for other types of questions.

This paper explores strategies for definition question processing to improve overall question answering performance. A key challenge we have to address is that of accurate recognition of such questions. Given an input question the Watson question analysis stage uses a definition question recognizer to detect this specific class of questions. We explore several approaches for recognition, including a rule based approach and a variety of statistical models.

Questions that are recognized as definition questions invoke search processes targeted towards dictionary-like sources in our system. We use a variety of such sources, such as standard English dictionaries, Wiktionary, WordNet, etc. After gathering supporting evidence for candidate answers extracted from these sources, our system routes the candidates to definition-specific candidate ranking models, which have been trained with selected feature sets.

The following sections present a description and evaluation of our approach for identifying and answering definition questions.

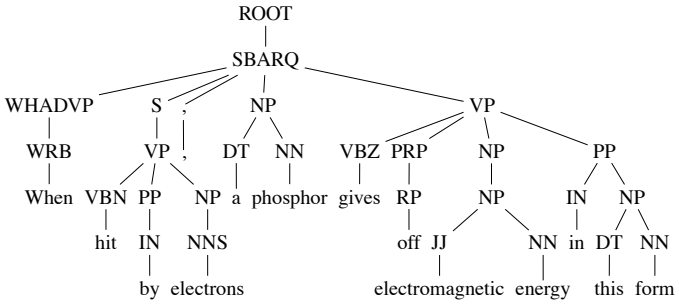

Figure 2: Constituency Tree

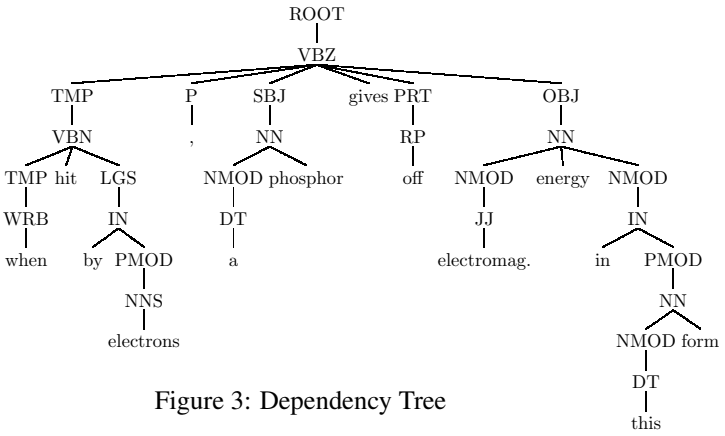

Figure 3: Dependency Tree

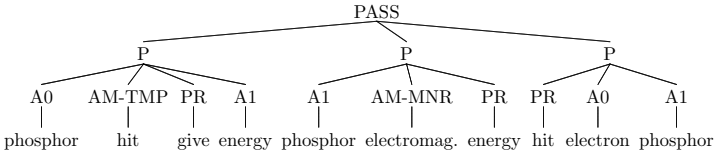

Figure 4: A tree encoding a Predicate Argument Structure Set

3 Kernel Models for Question Classification

Previous work (Zhang and Lee, 2003; Suzuki et al., 2003; Blunsom et al., 2006; Moschitti et al., 2007) as shown that syntactic structures are essential for QC. Given the novelty of both the domain and the type of our classification items, we rely on kernel methods to study and design effective representations. Indeed, these are excellent tools for automatic feature engineering, especially for unknown tasks and domains. Our approach consists of using SVMs and kernels for structured data applied to several types of structural lexical, syntactic and shallow semantic information.

3.1 Tree and Sequence Kernels

Kernel functions are implicit scalar products between data examples (i.e. questions in our case) in the very high dimensional space of substructures, where each of the latter is a component of the implicit vectors associated with the examples.

Although several kernels for structured data have been developed (see Section 6), the main distinctions in terms of feature spaces is given by the following three different kernels:

• Sequence Kernels (SK); we implemented the discontinuous string kernels described in (Shawe-Taylor and Cristianini, 2004). This allows for representing a string of symbols in terms of its possible substrings with gaps, i.e. an arbitrary number of symbols can be skipped during the generation of a substring. The symbols we used in the sequential descriptions of questions are words and part-of-speech tags (in two separate sequences). Consequently, all possible multiwords with gaps are features of the implicitly generated vector space.

• Syntactic Tree Kernel (STK) (Collins and Duffy, 2002) applied to constituency parse trees. This generates all possible tree fragments as features with the conditions that sibling nodes from the original trees cannot be separated. In other words, substructures are composed by atomic building blocks corresponding to nodes along with all their direct children. These, in case of a syntactic parse tree, are complete production rules of the associated parser grammar[2].

• Partial Tree Kernel (PTK) (Moschitti, 2006) applied to both constituency and dependency parse trees. This generates all possible tree fragments, as above, but sibling nodes can be separated (so they can be part of different tree fragments). In other words, a fragment is any possible tree path, from whose nodes other tree paths can depart. Consequently, an extremely rich feature space is generated. Of course, PTK subsumes STK but sometimes the latter provides more effective solutions as the number of irrelevant features is smaller as well.

When applied to sequences and tree structures, the kernels discussed above produce many different kinds of features. Therefore, the design of appropriate syntactic/semantic structures determines the representational power of the kernels. Hereafter, we show the models we used.

3.2 Syntactic Semantic Structures

We applied the above kernels to different structures. These can be divided in sequences of words (WS) and part of speech tags (PS) and different kinds of trees. For example, given the non-definition Jeopardy! question:

(5) GENERAL SCIENCE*: When hit by electrons, a phosphor gives off electromagnetic energy in this form.* (answer: light or photons),

we use the following sequences:

WS: [when][hit][by][electrons][,][a][phosphor][gives][off][electromagnetic][energy][in][this][form]

PS: [wrb][vbn][in][nns][,][dt][nn][vbz][rp][jj][nn][in][dt][nn]

Additionally, we use constituency trees (CTs), see Figure 2 and dependency structures converted into the dependency trees (DTs), e.g. shown in Figure 3. Note that, the POS-tags are central nodes, the grammatical relation label is added as a father node and all the relations with the other nodes are described by means of the connecting edges. Words are considered additional children of the POS-tag nodes (in this case the connecting edge just serves to add a lexical feature to the target POS-tag node).

Finally, we also use predicate argument structures generated by verbal and nominal relations according to PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004). Given the target sentence, the set of its predicates are extracted and converted into a forest, then a fake root node, PAS, is used to connect these trees. For example, Figure 4 illustrates a Predicate Argument Structures Set (PASS) encoding two relations, *give* and *hit*, as well as the nominalization *energy* along with all their arguments.

4 Experiments on Definition Question Classification

In these experiments, we study the role of kernel technology for the design of accurate classification of definition questions. We build several classifiers based on SVMs and kernel methods. Each classifier uses advanced syntactic/semantic structural features and their combination. We carry out an extensive comparison in terms of F1 between the different models on the Jeopardy! datasets.

4.1 Experimental Setup

Corpus: the data for our QC experiments consists of a randomly selected set of 33 Jeopardy! games[3]. These questions were manually annotated based on whether or not they are considered definitional. This resulted in 306 definition and 4964 non-definition clues. Each test set is stored in a separate file consisting of one line per question, which contains tab-separated clue information and the Jeopardy! category, e.g. INVEST in example (4).

Tools: for SVM learning, we used the SVMLight-TK software[4], which includes structural kernels in SVMLight (Joachims, 1999)[5]. For generating con-

[2]From here the name syntactic tree kernels

[3]Past Jeopardy! games can be downloaded from http://www.j-archive.com.

[4]Available at `http://dit.unitn.it/~moschitt`

[5]`http://svmlight.joachims.org`

stituency trees, we used the Charniak parser (Charniak, 2000). We also used the syntactic–semantic parser by Johansson and Nugues (2008) to generate dependency trees (Mel'čuk, 1988) and predicate argument trees according to the PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) frameworks.

Baseline Model: the first model that we used as a baseline is a rule-based classifier (RBC). The RBC leverages a set of rules that matches against lexical and syntactic information in the clue to make a binary decision on whether or not the clue is considered definitional. The rule set was manually developed by a human expert, and consists of rules that attempt to identify roughly 70 different constructs in the clues. For instance, one of the rules matches the parse tree structure for "It's X or Y", which will identify example (1) as a definition question.

Kernel Models: we apply the kernels described in Section 3 to the structures extracted from Jeopardy! clues. In particular, we design the following models: BOW, i.e. linear kernel on bag-of-words from the clues; WSK, PSK and CSK, i.e. SK applied to the word and POS-tag sequences from the clues, and the word sequence taken from the question categories, respectively; STK-CT, i.e. STK applied to CTs of the clue; PTK-CT and PTK-DT, i.e. PTK applied to CTs and DTs of the clues, respectively; PASS, i.e. PTK applied to the Predicate Argument Structure Set extracted from the clues; and RBC, i.e. a linear kernel applied to the vector only constituted by the 1/0 output of RBC.

Learning Setting: there is no particular parameterization. Since there is an imbalance between positive and negative examples, we used a Precision/Recall trade-off parameter in SVM-Light-TK equal to 5.[6]

Measures: the performance is measured with Precision, Recall and F1-measure. We estimated them by means of Leave-One-Out[7] (LOO) on the question set.

4.2 Results and Discussion

Table 1 shows the performance obtained using different kernels (feature spaces) with SVMs. We note that: first, RBC has good Recall but poor Precision. This is interesting since, on one hand, these results validate the complexity of the task: in order to capture the large variability of the positive examples, the rules developed by a skilled human designer are unable to be sufficiently precise to limit the recognition to those examples. On the other hand, RBC, being a rather different approach from SVMs, can be successfully exploited in a joint model with them.

Kernel Space	Prec.	Rec.	F1
RBC	28.27	70.59	**40.38**
BOW	47.67	46.73	47.20
WSK	47.11	50.65	48.82
STK-CT	50.51	32.35	39.44
PTK-CT	47.84	57.84	**52.37**
PTK-DT	44.81	57.84	50.50
PASS	33.50	21.90	26.49
PSK	39.88	45.10	42.33
CSK	39.07	77.12	**51.86**

Table 1: Kernel performance using leave-one-out cross-validation.

Second, BOW yields better F1 than RBC but it does not generalize well since its F1 is still low. When n-grams are also added to the model by means of WSK, the F1 improves by about 1.5 absolute points. As already shown in (Zhang and Lee, 2003; Moschitti et al., 2007), syntactic structures are needed to improve generalization.

Third, surprisingly with respect to previous work, STK applied to CT[8] provides accuracy lower than BOW, about 8 absolute points. The reason is due to the different nature of the Jeopardy! questions: large syntactic variability reduces the probability of finding general and well formed patterns, i.e. structures generated by entire production rules. This suggests that PTK, which can capture patterns derived from partial production rules, can be more effective. Indeed, PTK-CT achieves the highest F1, outperforming WSK also when used with a different syntactic paradigm, i.e. PTK-DT.

Next, PSK and PASS provide a lower accuracy but they may be useful in kernel combinations as they can complement the information captured by the other models. Interestingly, CSK alone is rather effective for classifying definition questions. We be-

[6] We have selected 5 as a reasonable value, which kept balanced Precision and Recall on a validation set.

[7] LOO applied to a corpus of N instances consists in training on $N-1$ examples and testing on the single held-out example. This process is repeated for all instances.

[8] Applying it to DT does not make much sense as already pointed out in (Moschitti, 2006).

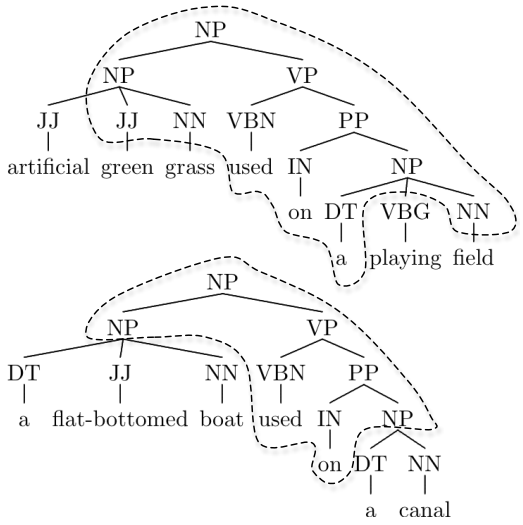

Figure 5: Similarity according to PTK and STK

lieve this is because definition questions are sometimes clustered into categories such as 4-LETTER WORDS or BEGINS WITH "B".

Moreover, we carried out qualitative error analysis on the PTK and STK outcome, which supported our initial hypothesis. Let us consider the bottom tree in Figure 5 in the training set. The top tree is a test example correctly classified by PTK but incorrectly classified by STK. The dashed line in the top tree contains the largest subtree matched by PTK (against the bottom tree), whereas the dashed line in the bottom tree indicates the largest subtree matched by STK (against the top tree). As the figure shows, PTK can exploit a larger number of partial patterns.

Finally, the above points suggest that different kernels produce complementary information. It is thus promising to experiment with their combinations. The joint models can be simply built by summing kernel functions together. The results are shown in Table 2. We note that: (i) CSK complements the WSK information, achieving a substantially better result, i.e. 62.95; (ii) PTK-CT+CSK performs even better than WSK+CSK (as PTK outperforms WSK); and (iii) adding RBC improves further on the above combinations, i.e. 68.11 and 67.32, respectively. This evidently demonstrates that RBC captures complementary information. Finally, more complex kernels, especially the overall kernel summation, do not seem to improve the performance. This is also confirmed by the PASS results derived in (Moschitti et al., 2007) on TREC QC.

Kernel Space	Prec.	Rec.	F1
WSK+CSK	70.00	57.19	62.95
PTK-CT+CSK	69.43	60.13	64.45
PTK-CT+WSK+CSK	68.59	62.09	**65.18**
CSK+RBC	47.80	74.51	58.23
PTK-CT+CSK+RBC	59.33	74.84	65.79
BOW+CSK+RBC	60.65	73.53	66.47
PTK-CT+WSK+CSK+RBC	67.66	66.99	**67.32**
PTK-CT+PASS+CSK+RBC	62.46	71.24	66.56
WSK+CSK+RBC	69.26	66.99	**68.11**
ALL	61.42	67.65	64.38

Table 2: Performance of Kernel Combinations using leave-one-out cross-validation.

5 Experiments on the Jeopardy System

Since the kernel-based classifiers perform substantially better than RBC, we incorporate the PTK-CT+WSK+CSK model[9] into Watson for definition classification and evaluated the QA performance against two baseline systems. For the end-to-end experiments, we used Watson's English Slot Grammar parser (McCord, 1980) to generate the constituency trees. The component level evaluation shows that we achieved comparable performance as previously discussed with ESG.

5.1 Experimental Setup

We integrated the classifier into the question analysis module, and incorporated additional components to search against dictionary resources and extract candidate answers from these search results when a question is classified as definitional. In the final machine learning models, a separate model is trained for definition questions to enable scoring tailored to the specific characteristics of those questions.

Based on our manually annotated gold standard, less than 10% of Jeopardy! questions are classified as definition questions. Due to their relatively low frequency we conduct two types of evaluations. The first is *definition-only evaluation*, in which we apply our definition question classifier to identify a large

[9]Since we aim to compare a purely statistical approach to the rule-based approach, we did not experiment with the model that uses RBC as a feature in our end-to-end experiments.

set of definition questions and evaluate the end-to-end system's performance on this large set of questions. These results enable us to draw statistically significant conclusions about our approach to addressing definition questions.

The second type of evaluation is *game-based evaluation*, which assesses the impact of our definition question processing on Watson performance while preserving the natural distribution of these question types in Jeopardy! data. Game-based evaluations situate the system's performance on definition questions relative to other types of questions, and enable us to gauge the component's contributions in a game-based setting.

For both evaluation settings, three configurations of Watson are used as follows:

- the **NoDef** system, in which Watson is configured without definition classification and processing, thereby treating all definition questions as regular factoid questions;
- the **StatDef** system, which leverages the statistical classifier and subsequent definition specific search and candidate generation components as described above; and
- the **RuleDef** system, in which Watson adopts RBC and employs the same additional definition search and candidate generation components as the StatDef system.

For the definition-only evaluation, we selected all questions recognized as definitional by the statistical classifier from roughly 1000 unseen games (60000 questions), resulting in a test set of 1606 questions. Due to the size of the initial set, it is impractical to manually create a gold standard for measuring Precision and Recall of the classifier. Instead, we compare the StatDef system against the NoDef on these 1606 questions using two metrics: accuracy, defined as the percentage of questions correctly answered, and p@70, the system's Precision when answering only the top 70% most confident questions. P@70 is an important metric in Jeopardy! game play as well as in real world applications where the system may refrain from answering a question when it is not confident about any of its answers. Since RBC identifies significantly more definition questions, we started with an initial set of roughly 300 games, from which the RBC identified 1875 questions as definitional. We compared the RuleDef system's performance on these questions against the NoDef baseline using the accuracy and p@70 metrics.

	NoDef	**StatDef**	**NoDef**	**RuleDef**
# Questions	1606	1606	1875	1875
Accuracy	63.76%	65.57%	56.64%	57.51%
P@70	82.22%	84.53%	72.73%	74.87%

Table 3: Definition-Only Evaluation Results

For the game-based evaluation, we randomly selected 66 unseen Jeopardy! games, consisting of 3546 questions after excluding audio/visual questions.[10] We contrast the StatDef system performance against that of NoDef and RuleDef along several dimensions: accuracy and p@70, described above, as well as earnings, the average amount of money earned for each game.

5.2 Definition-Only Evaluation

For the definition-only evaluation, we compared the StatDef system against the NoDef system on a set of 1606 questions that the StatDef system classified as definitional. The results are shown in the first two columns in Table 3. To contrast the gain obtained by the StatDef system against that achieved by the RuleDef system, we ran the RuleDef system over the 1875 questions identified as definitional by the rule-based classifier. We contrast the RuleDef system performance with that of the NoDef system, as shown in the last two columns in Table 3.

Our results show that based on both evaluation metrics, StatDef improved upon the NoDef baseline more than RuleDef improved on the same baseline system. Furthermore, for the accuracy metric where all samples are paired and independent, the difference in performance between the StatDef and NoDef systems is statistically significant at $p<0.05$, while that between the RuleDef and NoDef systems is not.

5.3 Game-Based Evaluation

The game-based evaluation was carried out on 66 unseen games (roughly 3500 questions). Of these

[10] Audio/visual questions are those accompanied by either an image or an audio clip. The text portions of these questions are often insufficient for identifying the correct answers.

	# Def Q's	Accuracy	P@70	Earnings
NoDef	0	69.71%	86.79%	$24,818
RuleDef	480	69.23%	86.31%	$24,397
StatDef	131	69.85%	87.19%	$25,109

Table 4: Game-Based Evaluation Results

questions, the StatDef system classified 131 of them as definitional while the RuleDef system identified 480 definition questions. Both systems were compared against the NoDef system using the accuracy, p@70, and earnings metric computed over all questions, as shown in Table 4.

Our results show that even though in the definition-only evaluation both the RuleDef and StatDef systems outperformed the NoDef baseline, in our game-based evaluation, the RuleDef system performed *worse* than the NoDef baseline. The lowered performance is due to the fact that the Precision of the RBC is much lower than that of the statistical classifier, and the special definition processing applied to questions that are erroneously classified as definitional was harmful. Our evaluation of this false positive set showed that its accuracy dropped by 6% compared to the NoDef system. On the other hand, the StatDef system outperformed the two other systems, and its accuracy improvement upon the RuleDef system is statistically significant at $p<0.05$.

6 Related Work

Our paper studies the use of advanced representation for QC in the Jeopardy! domain. As previously mentioned Jeopardy! questions are stated as affirmative sentences, which are different from the typical QA questions. For the design of our models, we have carefully taken into account previous work. This shows that semantics and syntax are essential to retrieve precise answers, e.g (Hickl et al., 2006; Voorhees, 2004; Small et al., 2004).

We focus on definition questions, which typically require more complex processing than factoid questions (Blair-Goldensohn et al., 2004; Chen et al., 2006; Shen and Lapata, 2007; Bilotti et al., 2007; Moschitti et al., 2007; Surdeanu et al., 2008; Echihabi and Marcu, 2003). For example, language models were applied to definitional QA in (Cui et al., 2005) to learn soft pattern models based on bigrams. Other related work, such as (Sasaki, 2005; Suzuki et al., 2002), was also very tied to bag-of-words features. Predicate argument structures have been mainly used for reranking (Shen and Lapata, 2007; Bilotti et al., 2007; Moschitti et al., 2007; Surdeanu et al., 2008).

Our work and methods are similar to (Zhang and Lee, 2003; Moschitti et al., 2007), which achieved the state-of-the-art in QC by applying SVMs along with STK-CT. The results were derived by experimenting with a TREC dataset[11](Li and Roth, 2002), reaching an accuracy of 91.8%. However, such data refers to typical instances from QA, whose syntactic patterns can be easily generalized by STK. In contrast, we have shown that STK-CT is not effective for our domain, as it presents very innovative elements: questions in affirmative and highly variable format. Thus, we employed new methods such as PTK, dependency structures, multiple sequence kernels including category information and many combinations.

Regarding the use of Kernel Methods, there is a considerably large body of work in Natural Language Processing, e.g. regarding syntactic parsing (Collins and Duffy, 2002; Kudo et al., 2005; Shen et al., 2003; Kudo and Matsumoto, 2003; Titov and Henderson, 2006; Toutanova et al., 2004), named entity recognition and chunking (Cumby and Roth, 2003; Daumé III and Marcu, 2004), relation extraction (Zelenko et al., 2002; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005; Zhang et al., 2005; Bunescu, 2007; Nguyen et al., 2009a), text categorization (Cancedda et al., 2003), word sense disambiguation (Gliozzo et al., 2005) and semantic role labeling (SRL), e.g. (Kazama and Torisawa, 2005; Che et al., 2006a; Moschitti et al., 2008).

However, ours is the first study on the use of several combinations of kernels applied to several structures on very complex data from the Jeopardy! domain.

7 Final Remarks and Conclusion

In this paper we have experimented with advanced structural kernels applied to several kinds of syntactic/semantic linguistic structures for the classification of questions in a new application domain, i.e. Jeopardy!. Our findings are summarized hereafter:

[11]Available at `http://cogcomp.cs.illinois.edu/Data/QA/QC/`

First, it should be noted that basic kernels, such as STK, PTK and SK, when applied to new representations, i.e. syntactic/semantic structures, constitute new kernels. Thus structural representations play a major role and, from this perspective, our paper makes a significant contribution.

Second, the experimental results show that the higher variability of Jeopardy! questions prevents us from achieving generalization with typical syntactic patterns even if they are derived by powerful methods such as STK. In contrast, partial patterns, such as those provided by PTK applied to constituency (or dependency) trees, prove to be effective.

In particular, STK has been considered as the best kernel for exploiting syntactic information in constituency trees, e.g. it is state-of-the-art in: QC (Zhang and Lee, 2003; Moschitti et al., 2007; Moschitti, 2008); SRL, (Moschitti et al., 2008; Moschitti et al., 2005; Che et al., 2006b); pronominal coreference resolution (Yang et al., 2006; Versley et al., 2008) and Relation Extraction (Zhang et al., 2006; Nguyen et al., 2009b). We showed that, in the complex domain of Jeopardy!, STK surprisingly provides low accuracy whereas PTK is rather effective and greatly outperforms STK. We have also provided an explanation of such behavior by means of error analysis: in contrast with traditional question classification, which focuses on basic syntactic patterns (e.g. "what", "where", "who" and "how"). Figure 5 shows that PTK captures partial patterns that are important for more complex questions like those in Jeopardy!

Third, we derived other interesting findings for NLP related to this novel domain, e.g.: (i) the impact of dependency trees is similar to the one of constituency trees. (ii) A simple computational representation of shallow semantics, i.e. PASS (Moschitti, 2008), does not work in Jeopardy!. (iii) Sequence kernels on category cues, i.e., higher level of lexical semantics, improve question classification. (iv) RBC jointly used with statistical approaches is helpful to tackle the Jeopardy! complexity.

Next, our kernel models improve up to 20 absolute percent points over n-grams based approaches, reaching a significant accuracy of about 70%. Watson, exploiting such a classifier, improved previous versions using RBC and no definition classification both in definition-only evaluations and in game-based evaluations.

Finally, we point out that:

- Jeopardy! has a variety of different special question types that are handled differently. We focus on kernel methods for definition question for two reasons. First, their recognition relies heavily on parse structures and is therefore more amenable to the approach proposed in the paper than the recognition of other question types. Second, definition is by far the most frequent special question type in Jeopardy!; therefore, we can obtain sufficient data for training and testing.

- We were unable to address the whole QC problem using a statistical model due to the lack of sufficient training data for most special question classes. Furthermore, we focused only on the definition classification and its impact on system performance due to space reasons.

- Our RBC has a rather imbalanced trade-off between Precision and Recall. This may not be the best operating point, but the optimal point is difficult to obtain empirically for an RBC, which is a strong motivation of the work in this paper. We experimented with tuning the trade-off between Precision and Recall with the RBC, but since RBC uses hand-crafted rules and does not have a parameter for that, ultimately the statistical approach proved more effective.

In future work, we plan to extend the current research by investigating models capable of exploiting predicate argument structures for question classification and answer reranking. The use of syntactic/semantic kernels is a promising research direction (Basili et al., 2005; Bloehdorn and Moschitti, 2007a; Bloehdorn and Moschitti, 2007b). In this perspective kernel learning is a very interesting research line, considering the complexity of representation and classification problems in which our kernels operate.

Acknowledgements

This work has been supported by the IBM's Open Collaboration Research (OCR) awards program. We are deeply in debt with Richard Johansson, who produced the earlier syntactic/semantic representations of the Jeopardy! questions from the text format.

References

Kisuh Ahn, Johan Bos, Stephen Clark, James R. Curran, Tiphaine Dalmas, Jochen L. Leidner, Matthew B. Smillie, and Bonnie Webber. 2004. Question answering with qed and wee at trec-2004. In E. M. Voorhees and L. P. Buckland, editors, *The Thirteenth Text REtrieval Conference, TREC 2004*, pages 595–599, Gaitersburg, MD.

Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2005. Effective use of WordNet semantics via kernel-based learning. In *Proceedings of CoNLL-2005*, pages 1–8, Ann Arbor, Michigan. Association for Computational Linguistics.

M. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. 2007. Structured retrieval for question answering. In *Proceedings of ACM SIGIR*.

S. Blair-Goldensohn, K. R. McKeown, and A. H. Schlaikjer. 2004. Answering definitional questions: A hybrid approach. In M. Maybury, editor, *Proceedings of AAAI 2004*. AAAI Press.

Stephan Bloehdorn and Alessandro Moschitti. 2007a. Combined syntactic and semantic kernels for text classification. In *Proceedings of ECIR 2007, Rome, Italy*.

Stephan Bloehdorn and Alessandro Moschitti. 2007b. Structure and semantics for expressive text kernels. In *In Proceedings of CIKM '07*.

Phil Blunsom, Krystle Kocik, and James R. Curran. 2006. Question classification with log-linear models. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 615–616, New York, NY, USA. ACM.

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT and EMNLP*, pages 724–731, Vancouver, British Columbia, Canada, October.

Razvan C. Bunescu. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of ACL*.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean Michel Renders. 2003. Word sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*.

Wanxiang Che, Min Zhang, Ting Liu, and Sheng Li. 2006a. A hybrid convolution tree kernel for semantic role labeling. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 73–80, Sydney, Australia, July. Association for Computational Linguistics.

Wanxiang Che, Min Zhang, Ting Liu, and Sheng Li. 2006b. A hybrid convolution tree kernel for semantic role labeling. In *Proceedings of the COLING/ACL on Main conference poster sessions*, COLING-ACL '06, pages 73–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Y. Chen, M. Zhou, and S. Wang. 2006. Reranking answers from definitional QA using language models. In *Proceedings of ACL*.

Charles Clarke, Gordon Cormack, and Thomas Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of the 24th SIGIR Conference*, pages 358–365.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL'02*.

H. Cui, M. Kan, and T. Chua. 2005. Generic soft pattern models for definitional QA. In *Proceedings of SIGIR*, Salvador, Brazil. ACM.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL*, pages 423–429, Barcelona, Spain, July.

Chad Cumby and Dan Roth. 2003. Kernel Methods for Relational Learning. In *Proceedings of ICML 2003*.

Hal Daumé III and Daniel Marcu. 2004. Np bracketing by maximum entropy tagging and SVM reranking. In *Proceedings of EMNLP'04*.

A. Echihabi and D. Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of ACL*.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3).

Danilo Giampiccolo, Pamela Froner, Anselmo Peñas, Christelle Ayache, Dan Cristea, Valentin Jijkoun, Petya Osenova, Paulo Rocha, Bogdan Sacaleanu, and Richard Suteliffe. 2007. Overview of the CLEF 2007 multilingual question anwering track. In *Proceedings of the Cross Language Evaluation Forum*.

Alfio Gliozzo, Claudio Giuliano, and Carlo Strapparava. 2005. Domain kernels for word sense disambiguation. In *Proceedings of ACL'05*, pages 403–410.

A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi, and B. Rink. 2006. Question answering with lcc chaucer at trec 2006. In *Proceedings of TREC*.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. *Advances in Kernel Methods – Support Vector Learning*, 13.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 183–187, Manchester, United Kingdom.

Michael Kaisser and Bonnie Webber. 2007. Question answering based on semantic roles. In *Proceedings of the Workshop on Deep Linguistic Processing*, DeepLP '07, pages 41–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jun'ichi Kazama and Kentaro Torisawa. 2005. Speeding up training with tree kernels for node relation labeling. In *Proceedings of HLT-EMNLP'05*.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL'03*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proceedings of ACL'05*.

Shyong Lam, David Pennock, Dan Cosley, and Steve Lawrence. 2003. 1 billion pages = 1 milllion dollars? mining the web to pay "who wants to be a millionaire? In *Proceedings of the 19th Conference on Uncertainty in AI*.

X. Li and D. Roth. 2002. Learning question classifiers. In *Proceedings of ACL*.

Michael C. McCord. 1980. Slot grammars. *Computational Linguistics*.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York, Albany.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, United States.

Alessandro Moschitti, Bonaventura Coppola, Ana-Maria Giuglea, and Roberto Basili. 2005. Hierarchical semantic role labeling. In *CoNLL 2005 shared task*.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In *Proceedings of ACL'07*.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of ECML'06*, pages 318–329.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *Proceeding of CIKM '08*, NY, USA.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009a. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of EMNLP*, pages 1378–1387, Singapore, August.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009b. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387, Morristown, NJ, USA. Association for Computational Linguistics.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Yutaka Sasaki, Chuan-Jie Lin, Kuang-hua Chen, and Hsin-Hsi Chen. 2007. Overview of the NTCIR-6 cross-lingual question answering (CLQA) task. In *Proceedings of the 6th NTCIR Workshop on Evaluation of Information Access Technologies*.

Y. Sasaki. 2005. Question answering as question-biased term extraction: A new approach toward multilingual qa. In *Proceedings of ACL*, pages 215–222.

John Shawe-Taylor and Nello Cristianini. 2004. *LaTeX User's Guide and Document Reference Manual*. Kernel Methods for Pattern Analysis, Cambridge University Press.

D. Shen and M. Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*.

L. Shen, A. Sarkar, and A. Joshi. 2003. Using LTAG Based Features in Parse Reranking. In *Proceedings of EMNLP*, Sapporo, Japan.

S. Small, T. Strzalkowski, T. Liu, S. Ryan, R. Salkin, N. Shimizu, P. Kantor, D. Kelly, and N. Wacholder. 2004. Hitiqa: Towards analytical question answering. In *Proceedings of COLING*.

M. Surdeanu, M. Ciaramita, and H. Zaragoza. 2008. Learning to rank answers on large online QA collections. In *Proceedings of ACL-HLT*, Columbus, Ohio.

J. Suzuki, Y. Sasaki, and E. Maeda. 2002. Svm answer selection for open-domain question answering. In *Proceedings of Coling*, pages 974–980.

Jun Suzuki, Hirotoshi Taira, Yutaka Sasaki, and Eisaku Maeda. 2003. Question classification using hdag kernel. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*, pages 61–68, Sapporo, Japan, July. Association for Computational Linguistics.

Ivan Titov and James Henderson. 2006. Porting statistical parsers with data-defined kernels. In *Proceedings of CoNLL-X*.

Kristina Toutanova, Penka Markova, and Christopher Manning. 2004. The Leaf Path Projection View of Parse Trees: Exploring String Kernels for HPSG Parse Selection. In *Proceedings of EMNLP 2004*.

Yannick Versley, Alessandro Moschitti, Massimo Poesio, and Xiaofeng Yang. 2008. Coreference systems based on kernels methods. In *The 22nd International Conference on Computational Linguistics (Coling'08)*, Manchester, England.

Ellen M. Voorhees and Hoa Trang Dang. 2006. Overview of the TREC 2005 question answering track. In *Proceedings of the TREC 2005 Conference*.

E. M. Voorhees. 2004. Overview of the trec 2004 question answering track. In *Proceedings of TREC 2004*.

Xiaofeng Yang, Jian Su, and Chewlim Tan. 2006. Kernel-based pronoun resolution with structured syntactic knowledge. In *Proc. COLING-ACL 06*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2002. Kernel methods for relation extraction. In *Proceedings of EMNLP-ACL*, pages 181–201.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM Press.

Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. 2005. Discovering relations between named entities from a large raw corpus using tree similarity-based clustering. In *Proceedings of IJCNLP'2005, Lecture Notes in Computer Science (LNCS 3651)*, pages 378–389, Jeju Island, South Korea.

Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring Syntactic Features for Relation Extraction using a Convolution tree kernel. In *Proceedings of NAACL*.