SPIED: Stanford Pattern-based Information Extraction and Diagnostics

Sonal Gupta **Christopher D. Manning**
Department of Computer Science
Stanford University
{sonal, manning}@cs.stanford.edu

Abstract

This paper aims to provide an effective interface for progressive refinement of pattern-based information extraction systems. Pattern-based information extraction (IE) systems have an advantage over machine learning based systems that patterns are easy to customize to cope with errors and are interpretable by humans. Building a pattern-based system is usually an iterative process of trying different parameters and thresholds to learn patterns and entities with high precision and recall. Since patterns are interpretable to humans, it is possible to identify sources of errors, such as patterns responsible for extracting incorrect entities and vice-versa, and correct them. However, it involves time consuming manual inspection of the extracted output. We present a light-weight tool, SPIED, to aid IE system developers in learning entities using patterns with bootstrapping, and visualizing the learned entities and patterns with explanations. SPIED is the first publicly available tool to visualize diagnostic information of multiple pattern learning systems to the best of our knowledge.

1 Introduction

Entity extraction using rules dominates commercial industry, mainly because rules are effective, interpretable by humans, and easy to customize to cope with errors (Chiticariu et al., 2013). Rules, which can be hand crafted or learned by a system, are commonly created by looking at the context around already known entities, such as surface word patterns (Hearst, 1992) and dependency patterns (Yangarber et al., 2000). Building a pattern-based learning system is usually a repetitive process, usually performed by the system developer, of manually examining a system's output to identify improvements or errors introduced by changing the entity or pattern extractor. Interpretability of patterns makes it easier for humans to identify sources of errors by inspecting patterns that extracted incorrect instances or instances that resulted in learning of bad patterns. Parameters range from window size of the context in surface word patterns to thresholds for learning a candidate entity. At present, there is a lack of tools helping a system developer to understand results and to improve results iteratively.

Visualizing diagnostic information of a system and contrasting it with another system can make the iterative process easier and more efficient. For example, consider a user trying to decide on the context's window size in surface words patterns. And the user deliberates that part-of-speech (POS) restriction of context words might be required for a reduced window size to avoid extracting erroneous mentions.[1] By comparing and contrasting extractions of two systems with different parameters, the user can investigate the cases in which the POS restriction is required with smaller window size, and whether the restriction causes the system to miss some correct entities. In contrast, comparing just accuracy of two systems does not allow inspecting finer details of extractions that increase or decrease accuracy and to make changes accordingly.

In this paper, we present a pattern-based entity learning and diagnostics tool, SPIED. It consists of two components: 1. pattern-based entity learning using bootstrapping (SPIED-Learn), and 2. visualizing the output of one or two entity learning systems (SPIED-Viz). SPIED-Viz is independent of SPIED-Learn and can be used with any pattern-based entity learner. For demonstration, we use the output of SPIED-Learn as an input to SPIED-

[1] A shorter context size usually extracts entities with higher recall but lower precision.

Viz. SPIED-Viz has pattern-centric and entity-centric views, which visualize learned patterns and entities, respectively, and the explanations for learning them. SPIED-Viz can also contrast two systems by comparing the ranks of learned entities and patterns. In this paper, as a concrete example, we learn and visualize drug-treatment (DT) entities from unlabeled patient-generated medical text, starting with seed dictionaries of entities for multiple classes. The task was proposed and further developed in Gupta and Manning (2014b) and Gupta and Manning (2014a).

Our contributions in this paper are: 1. we present a novel diagnostic tool for visualization of output of multiple pattern-based entity learning systems, and 2. we release the code of an end-to-end pattern learning system, which learns entities using patterns in a bootstrapped system and visualizes its diagnostic output. The pattern learning code is available at `http://nlp.stanford.edu/software/patternslearning.shtml`. The visualization code is available at `http://nlp.stanford.edu/software/patternviz.shtml`.

2 Learning Patterns and Entities

Bootstrapped systems have been commonly used to learn entities (Riloff, 1996; Collins and Singer, 1999). SPIED-Learn is based on the system described in Gupta and Manning (2014a), which builds upon the previous bootstrapped pattern-learning work and proposed an improved measure to score patterns (Step 3 below). It learns entities for given classes from unlabeled text by bootstrapping from seed dictionaries. Patterns are learned using labeled entities, and entities are learned based on the extractions of learned patterns. The process is iteratively performed until no more patterns or entities can be learned. The following steps give a short summary of the iterative learning of entities belonging to a class DT:

1. Data labeling: The text is labeled using the class dictionaries, starting with the seed dictionaries in the first iteration. A phrase matching a dictionary phrase is labeled with the dictionary's class.

2. Pattern generation: Patterns are generated using the context around the positively labeled entities to create candidate patterns for DT.

3. Pattern learning: Candidate patterns are scored using a pattern scoring measure and the top ones are added to the list of learned patterns for DT. The maximum number of patterns learned is given as an input to the system by the developer.

4. Entity learning: Learned patterns for the class are applied to the text to extract candidate entities. An entity scorer ranks the candidate entities and adds the top entities to DT's dictionary. The maximum number of entities learned is given as an input to the system by the developer.

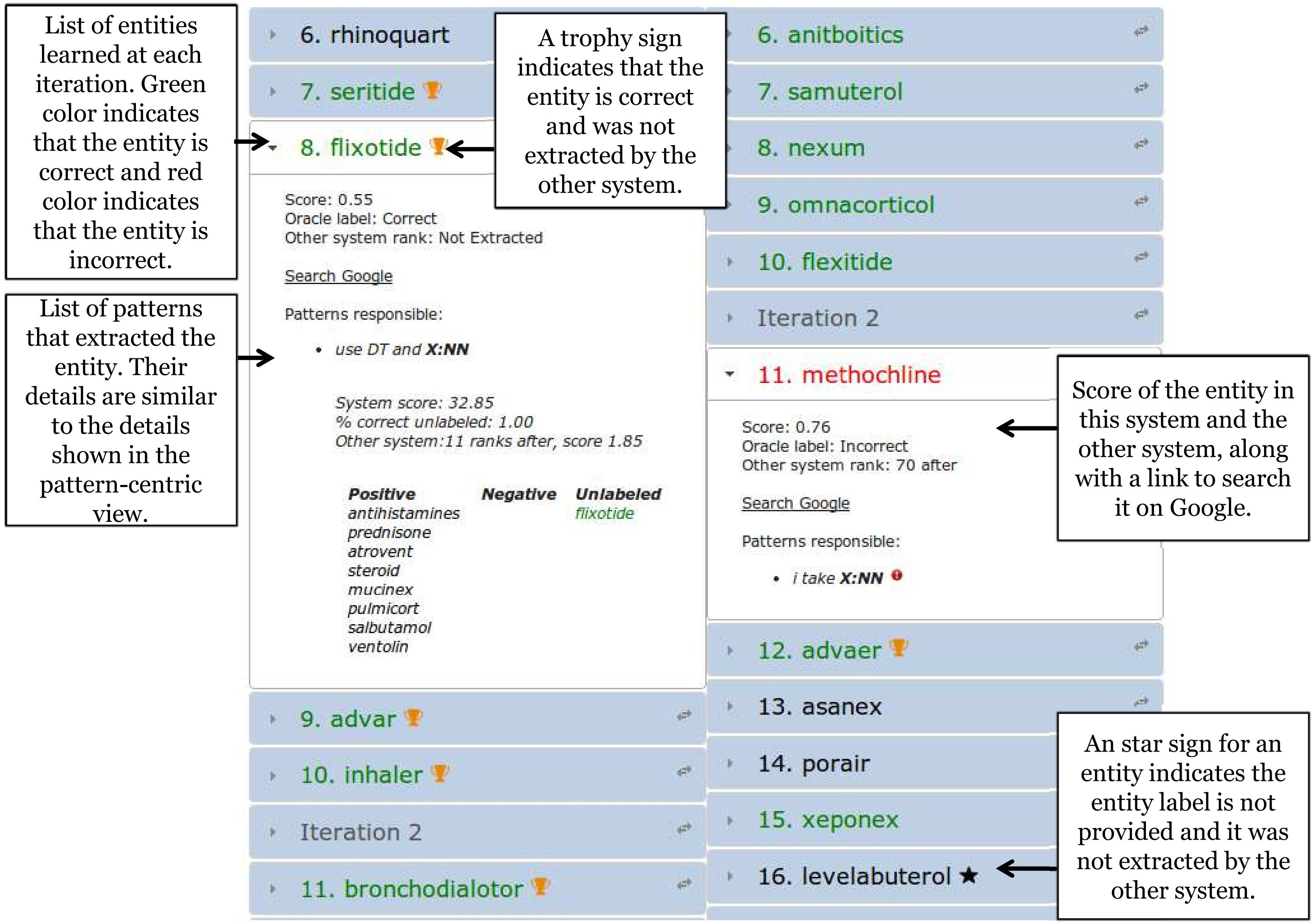
5. Repeat steps 1-4 for a given number of iterations.

SPIED provides an option to use any of the pattern scoring measures described in (Riloff, 1996; Thelen and Riloff, 2002; Yangarber et al., 2002; Lin et al., 2003; Gupta and Manning, 2014b). A pattern is scored based on the positive, negative, and unlabeled entities it extracts. The positive and negative labels of entities are heuristically determined by the system using the dictionaries and the iterative entity learning process. The oracle labels of learned entities are not available to the learning system. Note that an entity that the system considered positive might actually be incorrect, since the seed dictionaries can be noisy and the system can learn incorrect entities in the previous iterations, and vice-versa. SPIED's entity scorer is the same as in Gupta and Manning (2014a).

Each candidate entity is scored using weights of the patterns that extract it and other entity scoring measures, such as TF-IDF. Thus, learning of each entity can be explained by the learned patterns that extract it, and learning of each pattern can be explained by all the entities it extracts.

3 Visualizing Diagnostic Information

SPIED-Viz visualizes learned entities and patterns from one or two entity learning systems, and the diagnostic information associated with them. It optionally uses the oracle labels of learned entities to color code them, and contrast their ranks of correct/incorrect entities when comparing two systems. The oracle labels are usually determined by manually judging each learned entity as correct or incorrect. SPIED-Viz has two views: 1. a pattern-centric view that visualizes patterns of one


List of entities learned at each iteration. Green color indicates that the entity is correct and red color indicates that the entity is incorrect.

List of patterns that extracted the entity. Their details are similar to the details shown in the pattern-centric view.

A trophy sign indicates that the entity is correct and was not extracted by the other system.

Score of the entity in this system and the other system, along with a link to search it on Google.

An star sign for an entity indicates the entity label is not provided and it was not extracted by the other system.

6. rhinoquart
7. seritide
8. flixotide

Score: 0.55
Oracle label: Correct
Other system rank: Not Extracted

Search Google

Patterns responsible:

- *use DT and **X:NN***

System score: 32.85
% correct unlabeled: 1.00
Other system:11 ranks after, score 1.85

Positive	***Negative***	***Unlabeled***
antihistamines		*flixotide*
prednisone		
atrovent		
steroid		
mucinex		
pulmicort		
salbutamol		
ventolin		

9. advar
10. inhaler
Iteration 2
11. bronchodialotor

6. anitboitics
7. samuterol
8. nexum
9. omnacorticol
10. flexitide
Iteration 2
11. methochline

Score: 0.76
Oracle label: Incorrect
Other system rank: 70 after

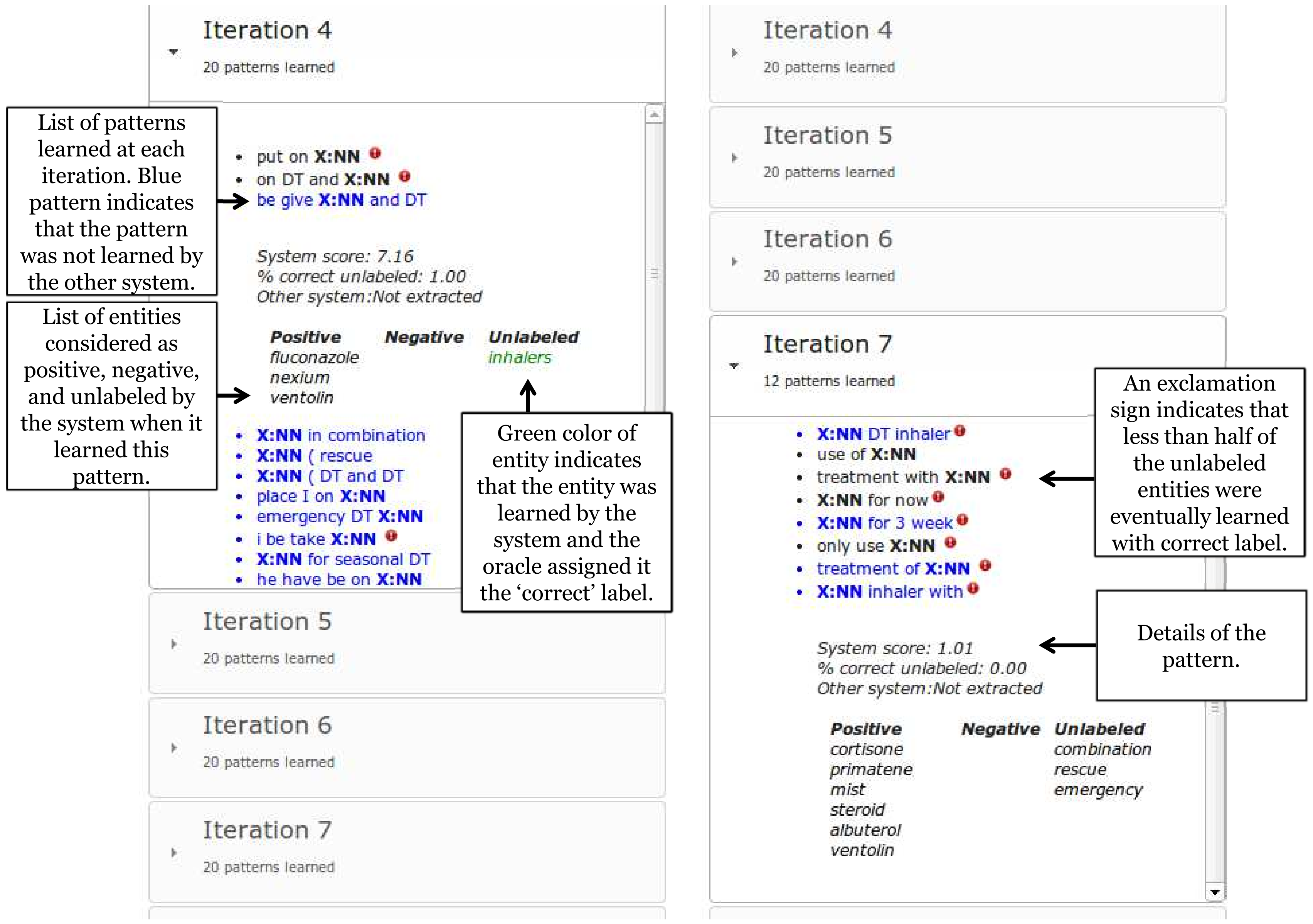
Search Google

Patterns responsible:

- *i take **X:NN***

12. advaer
13. asanex
14. porair
15. xeponex
16. levelabuterol ★

Figure 1: Entity centric view of SPIED-Viz. The interface allows the user to drill down the results to diagnose extraction of correct and incorrect entities, and contrast the details of the two systems. The entities that are not learned by the other system are marked with either a trophy (correct entity), a thumbs down (incorrect entity), or a star icon (oracle label missing), for easy identification.



Iteration 4

20 patterns learned

- put on **X:NN**
- on DT and **X:NN**
- be give **X:NN** and DT

System score: 7.16
% correct unlabeled: 1.00
Other system:Not extracted

Positive	***Negative***	***Unlabeled***
fluconazole		*inhalers*
nexium		
ventolin		

- **X:NN** in combination
- **X:NN** (rescue
- **X:NN** (DT and DT
- place I on **X:NN**
- emergency DT **X:NN**
- i be take **X:NN**
- **X:NN** for seasonal DT
- he have be on **X:NN**

Iteration 5

20 patterns learned

Iteration 6

20 patterns learned

Iteration 7

20 patterns learned

List of patterns learned at each iteration. Blue pattern indicates that the pattern was not learned by the other system.

List of entities considered as positive, negative, and unlabeled by the system when it learned this pattern.

Green color of entity indicates that the entity was learned by the system and the oracle assigned it the ‘correct’ label.

Iteration 4

20 patterns learned

Iteration 5

20 patterns learned

Iteration 6

20 patterns learned

Iteration 7

12 patterns learned

- **X:NN** DT inhaler
- use of **X:NN**
- treatment with **X:NN**
- **X:NN** for now
- **X:NN** for 3 week
- only use **X:NN**
- treatment of **X:NN**
- **X:NN** inhaler with

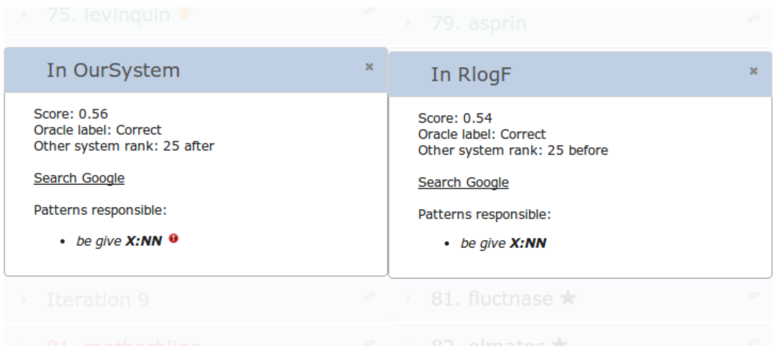
System score: 1.01
% correct unlabeled: 0.00
Other system:Not extracted

Positive	***Negative***	***Unlabeled***
cortisone		*combination*
primatene		*rescue*
mist		*emergency*
steroid		
albuterol		
ventolin		

An exclamation sign indicates that less than half of the unlabeled entities were eventually learned with correct label.

Details of the pattern.

Figure 2: Pattern centric view of SPIED-Viz.


In OurSystem

Score: 0.56
Oracle label: Correct
Other system rank: 25 after

Search Google

Patterns responsible:

- *be give* ***X:NN***

In RlogF

Score: 0.54
Oracle label: Correct
Other system rank: 25 before

Search Google

Patterns responsible:

- *be give* ***X:NN***

Figure 3: When the user click on the compare icon for an entity, the explanations of the entity extraction for both systems (if available) are displayed. This allows direct comparison of why the two systems learned the entity.

to two systems, and 2. an entity centric view that mainly focuses on the entities learned. Figure 1 shows a screenshot of the entity-centric view of SPIED-Viz. It displays following information:

Summary: A summary information of each system at each iteration and overall. It shows for each system the number of iterations, the number of patterns learned, and the number of correct and incorrect entities learned.

Learned Entities with provenance: It shows ranked list of entities learned by each system, along with an explanation of why the entity was learned. The details shown include the entity's oracle label, its rank in the other system, and the learned patterns that extracted the entity. Such information can help the user to identify and inspect the patterns responsible for learning an incorrect entity. The interface also provides a link to search the entity along with any user provided keywords (such as domain of the problem) on Google.

System Comparison: SPIED-Viz can be used to compare entities learned by two systems. It marks entities that are learned by one system but not by the other system, by either displaying a trophy sign (if the entity is correct), a thumbs down sign (if the entity is incorrect), or a star sign (if the oracle label is not provided).

The second view of SPIED-Viz is pattern-centric. Figure 2 shows a screenshot of the pattern-centric view. It displays the following information.

Summary: A summary information of each system including the number of iterations and number of patterns learned at each iteration and overall.

Learned Patterns with provenance: It shows ranked list of patterns along with the entities it extracts and their labels. Note that each pattern is associated with a set of positive, negative and unlabeled entities, which were used to determine its score.[2] It also shows the percentage of unlabeled entities extracted by a pattern that were eventually learned by the system and assessed as correct by the oracle. A smaller percentage means that the pattern extracted many entities that were either never learned or learned but were labeled as incorrect by the oracle.

Figure 3 shows an option in the entity-centric view when hovering over an entity opens a window on the side that shows the diagnostic information of the entity learned by the other system. This direct comparison is to directly contrast learning of an entity by both systems. For example, it can help the user to inspect why an entity was learned at an earlier rank than the other system.

An advantage of making the learning entities component and the visualization component independent is that a developer can use any pattern scorer or entity scorer in the system without depending on the visualization component to provide that functionality.

[2]Note that positive, negative, and unlabeled labels are different from the oracle labels, correct and incorrect, for the learned entities. The former refer to the entity labels considered by the system when learning the pattern, and they come from the seed dictionaries and the learned entities. A positive entity considered by the system can be labeled as incorrect by the human assessor, in case the system made a mistake in labeling data, and vice-versa.

4 System Details

SPIED-Learn uses TokensRegex (Chang and Manning, 2014) to create and apply surface word patterns to text. SPIED-Viz takes details of learned entities and patterns as input in a JSON format. It uses Javascript, angular, and jquery to visualize the information in a web browser.

5 Related Work

Most interactive IE systems focus on annotation of text, labeling of entities, and manual writing of rules. Some annotation and labeling tools are: MITRE's Callisto[3], Knowtator[4], SAPIENT (Liakata et al., 2009), brat[5], Melita (Ciravegna et al., 2002), and XConc Suite (Kim et al., 2008). Akbik et al. (2013) interactively helps non-expert users to manually write patterns over dependency trees. GATE[6] provides the JAPE language that recognizes regular expressions over annotations. Other systems focus on reducing manual effort for developing extractors (Brauer et al., 2011; Li et al., 2011). In contrast, our tool focuses on visualizing and comparing diagnostic information associated with pattern learning systems.

WizIE (Li et al., 2012) is an integrated environment for annotating text and writing pattern extractors for information extraction. It also generates regular expressions around labeled mentions and suggests patterns to users. It is most similar to our tool as it displays an explanation of the results extracted by a pattern. However, it is focused towards hand writing and selection of rules. In addition, it cannot be used to directly compare two pattern learning systems.

What's Wrong With My NLP?[7] is a tool for jointly visualizing various natural language processing formats such as trees, graphs, and entities. It can be used alongside our system to visualize the patterns since we mainly focus on diagnostic information.

6 Future Work and Conclusion

We plan to add a feature for a user to provide the oracle label of a learned entity using the interface. Currently, the oracle labels are assigned offline. We also plan to extend SPIED to visualize diagnostic information of learned relations from a pattern-based relation learning system. Another avenue of future work is to evaluate SPIED-Viz by studying its users and their interactions with the system. In addition, we plan to improve the visualization by summarizing the diagnostic information, such as which parameters led to what mistakes, to make it easier to understand for systems that extract large number of patterns and entities.

In conclusion, we present a novel diagnostic tool for pattern-based entity learning that visualizes and compares output of one to two systems. It is light-weight web browser based visualization. The visualization can be used with any pattern-based entity learner. We make the code of an end-to-end system freely available for research purpose. The system learns entities and patterns using bootstrapping starting with seed dictionaries, and visualizes the diagnostic output. We hope SPIED will help other researchers and users to diagnose errors and tune parameters in their pattern-based entity learning system in an easy and efficient way.

[3]`http://callisto.mitre.org`

[4]`http://knowtator.sourceforge.net`

[5]`http://brat.nlplab.org`

[6]`http://gate.ac.uk`

[7]`https://code.google.com/p/whatswrong`

References

Alan Akbik, Oresti Konomi, and Michail Melnikov. 2013. Propminer: A workflow for interactive information extraction and exploration using dependency trees. In *ACL (Conference System Demonstrations)*, pages 157–162.

Falk Brauer, Robert Rieger, Adrian Mocan, and Wojciech M. Barczynski. 2011. Enabling information extraction by inference of regular expressions from sample entities. In *CIKM*, pages 1285–1294.

Angel X. Chang and Christopher D. Manning. 2014. TokensRegex: Defining cascaded regular expressions over tokens. In *Stanford University Technical Report*.

Laura Chiticariu, Yunyao Li, and Frederick R. Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, pages 827–832.

Fabio Ciravegna, Alexiei Dingli, Daniela Petrelli, and Yorick Wilks. 2002. User-system cooperation in document annotation based on information extraction. In *In Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02*, pages 122–137. Springer Verlag.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empir-*

ical Methods in Natural Language Processing and Very Large Corpora, pages 100–110.

Sonal Gupta and Christopher D. Manning. 2014a. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning (CoNLL)*.

Sonal Gupta and Christopher D. Manning. 2014b. Induced lexico-syntactic patterns improve information extraction from online medical forums. *Under Submission*.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational linguistics*, COLING '92, pages 539–545.

Jin-Dong Kim, Tomoko Ohta, and Jun ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*.

Yunyao Li, Vivian Chu, Sebastian Blohm, Huaiyu Zhu, and Howard Ho. 2011. Facilitating pattern discovery for relation extraction with semantic-signature-based clustering. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1415–1424.

Yunyao Li, Laura Chiticariu, Huahai Yang, Frederick R. Reiss, and Arnaldo Carreno-fuentes. 2012. Wizie: A best practices guided development environment for information extraction. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 109–114.

Maria Liakata, Claire Q, and Larisa N. Soldatova. 2009. Semantic annotation of papers: Interface & enrichment tool (sapient). In *Proceedings of the BioNLP 2009 Workshop*, pages 193–200.

Winston Lin, Roman Yangarber, and Ralph Grishman. 2003. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of the ICML 2003 Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the 13th National Conference on Artificial Intelligence*, AAAI'96, pages 1044–1049.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '02, pages 214–221.

Roman Yangarber, Ralph Grishman, and Pasi Tapanainen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics*, COLING '00, pages 940–946.

Roman Yangarber, Winston Lin, and Ralph Grishman. 2002. Unsupervised learning of generalized names. In *Proceedings of the 19th International Conference on Computational Linguistics*, COLING '02.