# Integrating Logical Inference Into Statistical Text Classification Applications

**Andrew S. Gordon and Reid Swanson**

Institute for Creative Technologies, University of Southern California
13274 Fiji Way, Marina del Rey, CA 90292 USA
gordon@ict.usc.edu, swansonr@ict.usc.edu

**Abstract**

Contemporary statistical text classification is becoming increasingly common across a wide range of everyday applications. Typically, the bottlenecks in performance are the availability and consistency of large amounts of training data. We argue that these techniques could be improved by seamlessly integrating logical inference into the text encoding pipeline, making it possible to utilize large-scale commonsense and special-purpose knowledge bases to aid in the interpretation and encoding of documents.

## Statistical text classification is good, not great

Contemporary statistical text classification is becoming increasingly common across a wide range of everyday applications, used in cases where a label must be assigned to textual material based on changing or idiosyncratic characteristics. For example, statistical text classification is commonly used for email spam filtering, where previous judgments of spam are used as training data to create a fully automated spam classifier using machine-learning techniques. In general, the performance of these techniques is bounded by two factors. First, large amounts of training data are typically needed in order to achieve reasonable levels of performance. Second, the level of inter-rater agreement in the labeling of training data sets an upper limit on the possible performance that a machine-learning algorithm can achieve. Accordingly, research in statistical text classification focuses on improving learning curves, i.e. finding ways to achieve the best possible performance given the available training data.

Statistical text classification technology has a number of attractive characteristics, particularly when compared to knowledge-rich approaches that involve logical inference. For instance, knowledge-based techniques typically require substantial amounts of knowledge engineering before a single problem can be tackled. However, statistical techniques can yield some performance even with small amounts of training data. Also, knowledge-based techniques are often brittle in their ability to handle inconsistent knowledge. In contrast, statistical techniques simply perform with less accuracy in the face of inconsistent training data.

On the downside, statistical text classifiers can be extremely slow learners, particularly when it comes to generalizing over high-level relations. Ideally, my spam filter should realize that *any offer to sell me pharmaceuticals* is spam unless it is from my pharmacist and for my ailments, and it should be able to learn this from two or three examples. Instead, a shockingly large amount of training data is necessary to get the filter to recognize the synonymous ways of making an offer in the English language, the names of all of the varieties of pharmaceuticals that are out there, and the terms for the multitude of human ailments that these drugs are meant to address. No human would require this amount of instruction to perform this task, largely because of the ease in which we apply our background knowledge.

In this paper we propose that this downside can be moderated by seamlessly integrating logical inference into the text-encoding pipeline of a statistical text classifier. Our approach retains all of the advantageous qualities of statistical approaches, but improves performance by considering the logically inferred consequences of the text.

## Changing the text encoding pipeline

Our approach to integrating logical inference involves modifying the text encoding stage of a statistical text classifier. Normally, all of the word-level features (e.g. unigrams, bigrams) that appear in the corpus of training data are identified, and individual training examples are encoded as feature vectors with a class label assignment. Machine learning techniques are then used to identify the features that are maximally predictive of the label given a new, unlabeled feature vector encoding of a test instance. In our approach, additional features are added (alongside word-level features) that correspond to the logical inferences that can be drawn from the text, which can then be considered by the machine-learning algorithm when determining the statistical regularities that exist in the learning space. In order to generate these additional inferential features from training text and test instances, three additional stages are necessary, as follows:

1. *Conversion into logical form*. The first step is to convert the text (training and test instances) into a logic-

form representation. While converting text into logic-form is a longstanding problem in Artificial Intelligence, contemporary techniques have made substantial progress using a combination of automated parsing and semantic role-labeling methods. In one method, input text documents can be segmented into sentences and parsed into a constituency parse trees (Charniak, 2000). These parse trees can then be represented as atoms in first-order logic using automated semantic-role labeling (Palmer et al., 2005), where the head verb of the parse tree is treated as the predicate and constituents of the parse tree are assigned as existentially quantified terms. Moldovan et al. (2003) describe an alternate method of logic-form conversion where all input words are treated as predicates over a set of existentially quantified terms that are combined using parsing axioms and abductive inference.

2. *Logical inference using knowledge bases*. Given a logic-form representation of the text (e.g. atoms with existentially quantified terms), traditional methods of automated logical inference can be utilized. Commonsense knowledge bases (e.g. Lenat, 1995) can be used to infer a set of commonsense consequences of the input, or special-purpose knowledge bases can be used to infer consequences that are specific to the classification task domain. In the simplest case, a knowledge base of axioms is combined with the conjunction of atoms generated from the input text, and then given as input to a first-order reasoning engine (e.g. Kalman, 2001). The output formulas are then collected as the set of inferred consequences.

3. *Encoding logical consequences as features*. Finally, the inferred consequences are encoded as a set of features to be included in the feature vector representation of the input text (alongside word-level features). Converting output formulas into flat feature lists can be done in a variety of different ways, each preserving different characteristics of the structured logical form. For example, bigrams of *Predicate+Word* can be generated for each of the words in the constituents that are used as existentially quantified terms in the inferred consequences, preserving information about which predications govern particular words in the input. Alternatively, the inferred consequences could be represented simply as a count of the number of times each axiom in the knowledge base fired given the input encoding (with one frequency-count feature per axiom), preserving the profile of the inference trace.

This three-stage approach changes only one aspect of a traditional statistical text classification system, namely how the training and test cases are encoded before they are presented to the machine-learning algorithm. By combining word-level features with those that can be inferred using logical inference, this approach places no strong performance requirements for any of the three challenging steps described above. If none of these three steps are implemented in a high-performing manner, then the resulting features are not going to be more predictive of the output label than the word-level features, and this will be evident to the machine-learning algorithm in its analysis of the feature space. Performance should be roughly equivalent to that which can be achieved without logical inference. However, if instead each of these three steps can be implemented in a high-performing manner, then the inferred features should by highly predictive, and adding these features will yield improved learning curves.

## The challenges of scale

The main challenge in implementing a high-performing system for generating inferential features is one of scale. Regardless of which scheme is used to encode logical consequences as feature vectors, the knowledge base that is used to produce these features must be of sufficient breadth and depth to add information to the representation of a text that cannot be trivially learned through word frequency statistics. Likewise, for every one of the predications that serve as antecedents in this knowledge base, sufficiently large amounts of linguistic information needs to be encoded to enable the accurate conversion of English sentences into logical form, e.g. large corpora of text with labeled predicate-argument relations (Palmer et al., 2005). In order to scale up in a tractable manner, it makes sense to pursue both resources in parallel, annotating only the linguistic forms of predications that correspond to axioms that exist in the knowledge base. This would, in effect, concentrate the labor of corpus annotation on forms for which some useful inferences can be made. As new commonsense or task-specific knowledge bases are created, these annotations would enable their integration into a text-encoding pipeline, with performance gains proportional to the relevance of the inferences.

## Acknowledgments

The project or effort depicted was sponsored by the U. S. Army Research, Development, and Engineering Command (RDECOM). The content or information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

## References

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

Kalman, J. 2001. *Automated Reasoning with Otter*. Paramus, NJ: Rinton Press.

Lenat, D. 1995. Cyc: A Large-Scale Investment in Knowledge Infrastructure. *Comm. of the ACM 38*(11).

Moldovan, D., Clark, C., Harabagiu, H. & Maiorano, S. 2003. COGEX: A Logic Prover for Question Answering, *Proceedings of HLT-NAACL-2003*.

Palmer, M., Gildea D., & Kingsbury, P. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics 31*(1).