LREC 2014

IllinoisCloudNLP: Text Analytics Services in the Cloud

Hao Wu, Zhiye Fei, Aaron Dai, Stephen Mayhew, Mark Sammons, Dan Roth
Department of Computer Science,
University of Illinois, Urbana-Champaign.
{haowu4,zfei2,dai13,mayhew2,mssammon,danr}@illinois.edu

Abstract

Natural Language Processing (NLP) continues to grow in popularity in a range of research and commercial applications. However, installing, maintaining, and running NLP tools can be time consuming, and many commercial and research end users have only intermittent need for large processing capacity. This paper describes IllinoisCloudNLP, an on-demand framework built around NLPCurator and Amazon Web Services' Elastic Compute Cloud (EC2). This framework provides a simple interface to end users via which they can deploy one or more NLPCurator instances on EC2, upload plain text documents, specify a set of Text Analytics tools (NLP annotations) to apply, and process and store or download the processed data. It also allows end users to use a model trained on their own data: IllinoisCloudNLP takes care of training, hosting, and applying it to new data just as it does with existing models within NLPCurator. As a representative use case, we describe our use of IllinoisCloudNLP to process 3.05 million documents used in the 2012 and 2013 Text Analysis Conference Knowledge Base Population tasks at a relatively deep level of processing, in approximately 20 hours, at an approximate cost of US$500; this is about 20 times faster than doing so on a single server and requires no human supervision and no NLP or Machine Learning expertise.

Keywords: Natural Language Processing Tools, Text Analytics, Cloud Computing

1. Motivation

Natural Language Processing (NLP) is an active research area, and its use in commercial enterprises has ballooned with the advent of the recent surge of interest in analyzing large amounts of data to support business intelligence practices. Many end users are small research teams or small- to medium-sized commercial enterprises with *intermittent processing loads*, i.e. they do not constantly process a large volume of data, but instead wish to process data as needed. When the need arises, they may have a large amount of data that they wish to process quickly. This use case has led to the commercial success of cloud services such as Amazon Web Services' Elastic Compute Cloud (EC2): a knowledgeable user can select from a wide variety of virtual machine images, start multiple instances on a group of EC2 host machines, and process data in parallel to achieve fast, high-volume data processing.

However, the use of Text Analytics tools and of cloud computing resources still requires significant end user expertise. While a number of NLP-related commercial and research efforts are underway to provide frameworks to support machine learning and NLP on cloud infrastructure (see Section 7.), many still require significant end user expertise and a corresponding investment of time, effort, and money.

IllinoisCloudNLP is a prototype cloud-based service that aims simply to provide scalable Text Analytics processing capabilities to non-expert end users, performing almost the entire set-up and the processing based on end user instructions via a simple application interface. IllinoisCloudNLP builds on NLPCurator, an NLP component management framework (Clarke et al., 2012). Clients sign up for an Amazon EC2 account, install a small client program, and can then upload and process text documents on multiple EC2 nodes. The interface manages the initialization of the nodes and the processing of the documents. It provides a straightforward interface for choosing the level of processing needed (which tools to run on the data) and for monitoring progress and node health. Once the documents are processed, the user can easily download or store the data, and terminate the EC2 nodes. The interface provides estimates of processing time and cost based on the user's data collection and choice of NLP annotations. The IllinoisCloudNLP infrastructure therefore allows the user to process documents *on demand*, and requires no NLP, Machine Learning, or Cloud Computing expertise.

This paper describes the analytics and classification capabilities that IllinoisCloudNLP currently provides (Sec. 2. and 3.), the system infrastructure that manages the Amazon servers (Sec. 4.), and the workflow from the user perspective (Sec. 5.). It also gives an example application of the IllinoisCloudNLP system (Sec. 6.), a comparison to some other NLP/Machine Learning frameworks (Sec. 7.) and our plans to improve the system (Sec. 8.).

2. NLPCurator

NLPCurator (Clarke et al., 2012) is an NLP management system that allows multiple NLP components to be distributed across multiple machines. End users can configure pipelines over components by specifying input dependencies for components in a configuration file. NLPCurator caches processed documents, allowing fast retrieval when multiple clients need to process the same data, or when a single user needs to repeat the same processing of the data (e.g. in an experimental setting). NLPCurator provides a single point of access to all these services, and a programmatic interface that allows end users to request and access NLP annotations within applications. It is complemented by Edison, a Java library that provides a large suite of NLP data structures and supports feature extraction and common experimental NLP tasks. Together, Edison and NLPCurator provide a straightforward API for applying a suite of state-of-the-art Illinois NLP tools to plain text documents.

IllinoisCloudNLP builds on this infrastructure and provides users with the ability to process text with a range of state-of-the-art tools including tokenization, Part of Speech (POS) tagging (Roth and Zelenko, 1998), shallow parsing (Punyakanok and Roth,

2001), Named Entity recognition (Ratinov and Roth, 2009), and Wikification (Cheng and Roth, 2013) (see `http://cogcomp.cs.illinois.edu/page/[software,demos]` for the tools named above, and others that we plan to integrate).
NLPCURATOR takes plain text as input, and generates data structures representing layers of annotation over the input text as output. The POS tagger and shallow parser use the relevant Penn Treebank tag sets. The Named Entity Recognizer labels proper nouns, either with the widely used CoNLL 2003 shared task scheme (Tjong Kim Sang and De Meulder, 2003), which has four categories: Person, Organization, Location, and Miscellaneous; or with the 18 entity types in the OntoNotes scheme (Hovy et al., 2006). The Wikifier identifies concepts in text, disambiguates them and grounds them by mapping the corresponding phrase to the most specific appropriate page in Wikipedia. The output data structures can be used directly, or written to a json or a binary format for disk storage.

3. ILLINOISCLASSIFIER

In addition to NLP annotation services, ILLINOISCLOUDNLP provides users with access to a generic classification architecture built around Learning Based Java (Rizzolo and Roth, 2010)[1]: the ILLINOISCLASSIFIER applies a suite of feature extractors and is trained simply by providing a set of plain-text documents which have been labeled according to the user's desired categories. ILLINOISCLASSIFIER uses this data to build a statistical model that, given a new set of previously unseen documents, assigns them labels seen during training. Labels can be associated with documents (e.g., categorizing them to topics), to sentences, or to other text snippets. Providing a training set with a different labeling scheme results in a different model. Crucially, the ILLINOISCLASSIFIER application is *agnostic to the actual task*, and is therefore easily applied to different document classification tasks, e.g. spam vs. not-spam; subject (sports vs. music vs. politics); author; etc.
The initial release of ILLINOISCLOUDNLP supports classification of spans of text, with the assumption that each span of text is an independent paragraph, sentence, or document, and uses simple n-gram features. Future releases will extend the expressiveness of the classification task and the feature space (see Section 8.).
We characterize the learning problem in the following way: the user wishes to learn a classifier $\mathcal{C}$ to assign one of a set of labels $\mathcal{L}_1, \mathcal{L}_2, ...\mathcal{L}_k \in \mathcal{L}$ to each text snippet $\mathcal{T}_i \in \mathcal{T}$, where $\mathcal{T}$ is the domain of text snippets that are suitable for such labels. For example, $\mathcal{T}$ could be newspaper articles, and $\mathcal{L}$ a set of subject labels such as "Sport", "Politics", "Finance" etc. Alternatively, $\mathcal{T}$ could be customer reviews of hotels, and $\mathcal{L}$ the associated rating from 1 to 5.
In the supervised learning setting used by ILLINOISCLASSIFIER, the user must have a sufficiently large set of text snippets that have been labeled with the target categories; this set of labeled data is typically split into *training* and *evaluation* data sets. The learning system generates an abstract representation of the text snippets and uses the correspondences of elements of this representation with target labels in the training data to automatically generate a model that can map unseen instances drawn from $\mathcal{T}$ to the target labels. The performance of the resulting model, which is usually measured on the held-out evaluation data, depends on the expressiveness of the abstract representation and the amount of labeled data available (Kearns and Vazirani, 1994). The learned model should perform as well on new snippets drawn from the same domain $\mathcal{T}$, even though it has not seen them before. It can therefore be used to label new data drawn from the same sources as the training data; for example, if you use a set of documents drawn from news articles labeled by subject, your trained classifier can be used to label new news articles with those subjects. The classifier considers multiple sets of features and uses cross-validation to choose the most appropriate one, without any user involvement.

[1] `http://cogcomp.cs.illinois.edu/page/software_view/LBJ`

4. ILLINOISCLOUDNLP Infrastructure

ILLINOISCLOUDNLP is a framework that allows non-expert end users to specify a text collection to be processed and a set of tools (which we will call here "NLP annotations") to be applied to it; it then generates a set of worker nodes on Amazon's EC2 to process the data. The end user must have their own Amazon EC2 account, which is used to pay the cost of processing the data.
The software package used by ILLINOISCLOUDNLP users is a lightweight application that controls and monitors a complete infrastructure built on Amazon Web Service (AWS). This package allows the user to manage an NLP service in a simple, user-friendly way. All NLP services that we provide are run on the Cloud, so the user doesn't have to install a large suite of software on their site. In the rest of this section we briefly describe the infrastructure underlying the system.

4.1. Amazon Web Services

Amazon Web Services (2014) (AWS) is a collection of cloud based infrastructure and services provided by Amazon. AWS gives users the ability to compute, store and manage their data on the cloud. Amazon charges the user based on the resources they choose to use: in effect, the user *"rents"* some of AWS's servers only for the time needed to process their data plus an additional charge to store data on the AWS servers and to access it. A specific example is given in Section 6.

4.2. Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (EC2) is an elastic infrastructure provided by AWS to support computing on demand. It allows users to specify a profile of one or more virtual computers for their use and pay hourly only for those resources they use. The user can then start and stop these machines as they wish, and when these Amazon Machine Instances (AMIs) are active, the user can access them either by connecting to them (e.g. via ssh) or using the Amazon API to programmatically interact with them. An alternative model is to install service software on these AMIs that can be run when they are active, and then have other applications use these services to handle CPU- or memory-intensive tasks. This framework provides a very economical way to support on-demand NLP services, since most state of the art NLP systems require a relatively large amount of RAM and processing power. In ILLINOISCLOUDNLP, all annotation processing is performed in EC2.

4.3. Amazon Simple Storage Service

Amazon Simple Storage Service (S3) provides cloud based key-value pair storage. Keys (created by the user) are identifiers associated with some piece of data, while a value can be an arbitrary data structure representing a document or, in this case, a set of annotations over a document. A user can create collections of key-value pairs (buckets) for their own use. Amazon charges a fee for storage and for accessing the data; at the time of writing, these fees are quite low. ILLINOISCLOUDNLP uses S3 to store processed data for retrieval by the user and/or later use on Amazon EC2.

4.4. ILLINOISCLOUDNLP Cloud Infrastructure

ILLINOISCLOUDNLP's NLP services are run as sets of AMIs (*"clusters"*) on EC2. Each curator cluster has three components:

- NLPCURATOR **Worker**. A number of NLPCURATOR workers receive jobs (documents to be processed) from the job queue, annotate the given document, and store the processed result. Each worker is an AMI that runs an instance of NLPCURATOR with a suite of Illinois NLP tools or a model trained using an ILLINOISCLASSIFIER component that is applied just like other NLPCURATOR components.
- TRAINING UNIT. A component that trains a classifier based on the data given by user. It stores a trained model to Amazon S3, so that NLPCURATOR **Worker**s can load it from S3 and run it on data.
- **Manager**. A control node that starts and stops **Workers** and TRAINING UNITs. It runs a central queue that stores all incoming jobs, which are then transferred to **Workers** or TRAINING UNITs.
- **Shared Data Store**. A shared data store, Amazon's S3 service, that is accessed by all ILLINOISCLOUDNLP components.

Figure 1 shows this architecture. The user workflow for ILLINOISCLOUDNLP is described in Section 5.

The design challenges underlying this architecture are how to manage the workload to optimize throughput and overall cost; how to handle problems that arise with the software being run on individual EC2 nodes; how to maintain user security; and how to balance ease of use with user control over the process.

To maximize ease of use, ILLINOISCLOUDNLP manages the workload dynamically. It initially starts five worker nodes to process the client's data, then monitors the throughput at the Job Queue component: if this is too slow, it starts another five worker nodes. It will repeat this process until a preset cap is reached (currently, 20 nodes, which is the standard limit imposed by Amazon on users). Once the job queue is exhausted, the worker nodes are shut down as they complete their tasks.

Some ill-formed text or extremely long documents can cause individual NLP annotators to fail. ILLINOISCLOUDNLP tries up to 5 times to annotate each document, then gives up. No annotations will be generated for these specific documents, but there is no effect on other documents. If a node should fail completely for some reason, a replacement node will be started automatically.

The ILLINOISCLOUDNLP interface provides a cost estimator pane which is periodically updated based on current system throughput. The interface also provides a "stop the cluster" button, so that at any time if a user decides to terminate an ongoing annotation task they can do so. In the event of a partially completed task, all completed annotations will be stored on S3. The user can also use the Amazon Web Service EC2 interface to stop nodes, or to check that nodes have been started.

For security and privacy, ILLINOISCLOUDNLP is set up for users to deploy it from their own Amazon Web Service account. The user logs into the ILLINOISCLOUDNLP's EC2 manager using substrings from their own security keys, which prevents other parties from accessing their EC2 nodes and their data.

5. Using ILLINOISCLOUDNLP

This section gives details on setting up ILLINOISCLOUDNLP and describes the workflow.

5.1. Prerequisites

The user must first set up a user account on Amazon Web Services' EC2 and S3 services. The account id and password will be used with the ILLINOISCLOUDNLP client to initiate document processing. It is also necessary to select and start an AMI (virtual machine) instance so that it is available when the client software connects to Amazon. The Java Development Kit (version 1.5 or higher) must be installed, and the *JAVA_HOME* and *JAVAC* environment variables must be set. The data to be processed must be plain text.

Installing the ILLINOISCLOUDNLP client is straightforward – visiting the software's home page[2] and downloading the zip file, unpacking the zip file in the desired location on the host machine, and running the start script.

5.2. Overview

Throughout this section, we use the term *annotations* to refer to the enriched output produced by NLPCURATOR, including models previously trained in the ILLINOISCLOUDNLP environment by the user.

The start script opens a locally hosted login page in a web browser. After the user enters their Amazon login credentials they indicate the data they wish to use, either selecting an S3 bucket they generated in a previous session or naming a new one. This allows users to simply access data they previously annotated without starting any annotation servers. (The user can decide to use a different data bucket or upload a new data set.)

[2] http://cogcomp.cs.illinois.edu/page/software_view/IllinoisCloudNLP

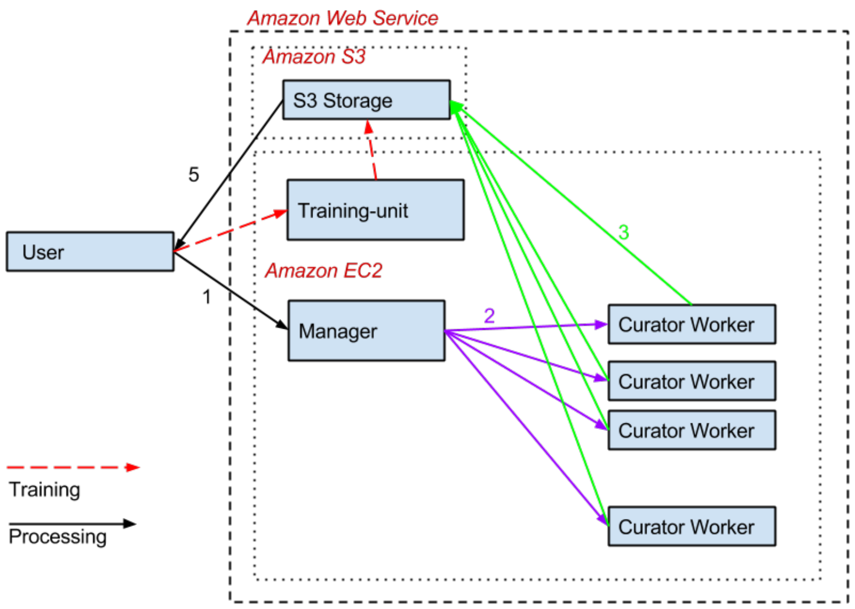


Figure 1: System architecture of ILLINOISCLOUDNLP. Arrows indicate workflow: 1. The client uploads documents to be processed or to train a classifier. 2. For an annotation task, the manager node transmits documents to worker nodes for processing. 3. Worker nodes store annotated documents using Amazon's S3 service. 4. When a model is trained, the client starts a training node, which stores the model it learns on S3. 5. The user can access the data on S3 without starting a cluster.

The interface then starts the manager component on the cloud. Once this has loaded, the user is directed to a browser-based interface which presents four options: 1) View existing annotations generated in previous runs. 2) Add new annotations to data previously uploaded and processed by the user. 3) Annotate new documents provided by the user with Illinois NLP services and/or user-trained classifiers. 4) Train a new classifier (see Figure 2). At any time after one or more jobs have been started, the user can visit the cost estimation pane to get an estimate of the expected time and cost. These estimates are dynamically updated as the task progresses.

5.3. Annotating Text

The user defines an annotation task by specifying a set of desired NLP annotations (see Figure 3). The list of available annotation components will also include any models trained by the user.
When the user is satisfied with their task specification, they start the process. Our client program will run the entire system as described in the previous sections, and push the raw documents into the job queue; all of this is initiated by a single click.
The *Jobs* pane displays information about ongoing annotation and training tasks, indicating which stage of processing the system has reached (an example is shown in Figure 4). Once the task is finished, the user is alerted and the annotations are stored in an S3 bucket with a name provided by the user. These annotations can be viewed or downloaded, and by default are stored on S3 for later use.

5.4. Using ILLINOISCLASSIFIER to Train a Model

Users can provide labeled text data to the ILLINOISCLASSIFIER component to train and evaluate a classifier (see Section 3.). Once data is provided, ILLINOISCLASSIFIER automatically takes care of the rest – dividing the input data into training and evaluation sets and tuning training parameters to learn a robust model. In the initial release, the ILLINOISCLASSIFIER uses simple n-gram features. In later releases, we will enrich the feature representation to use the other NLP annotations provided by the Annotation components.

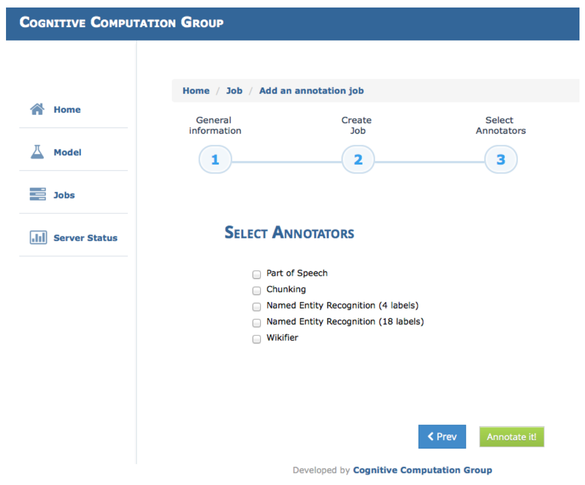

COGNITIVE COMPUTATION GROUP
Home
Model
Jobs
Server Status
Home / Job / Add an annotation job
General information
Create Job
Select Annotators
1
2
3
SELECT ANNOTATORS
Part of Speech
Chunking
Named Entity Recognition (4 labels)
Named Entity Recognition (18 labels)
Wikifier
‹ Prev
Annotate it!
Developed by Cognitive Computation Group

Figure 3: Selecting annotation components for an annotation task

To train a model using ILLINOISCLASSIFIER, the user first creates on their own machine a directory for the data $\mathcal{T}$ with a subdirectory for each label $\mathcal{L}_i \in \mathcal{L}$, each named for the target label, and containing the text snippets associated with that label. For example, for a spam filter, the user would use two sets of files, one for the spam and one for the non-spam. They would put each set of files in its own

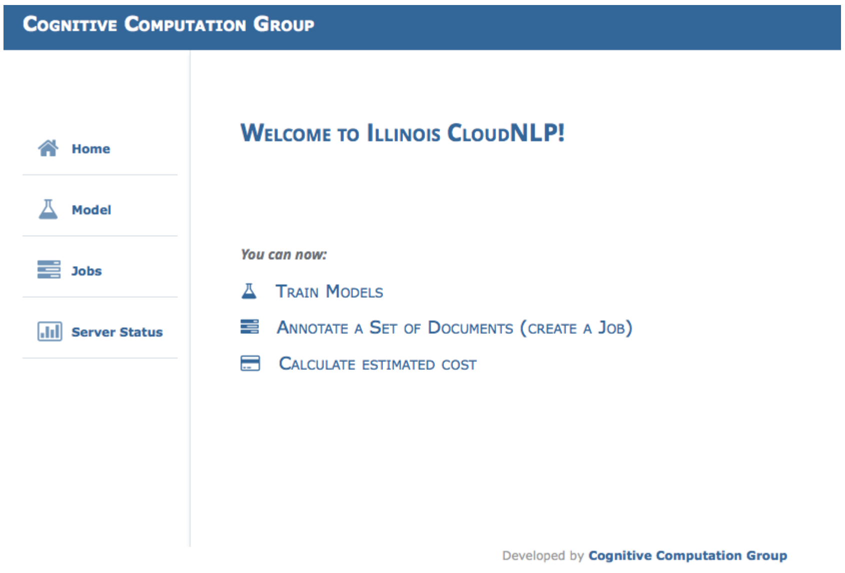
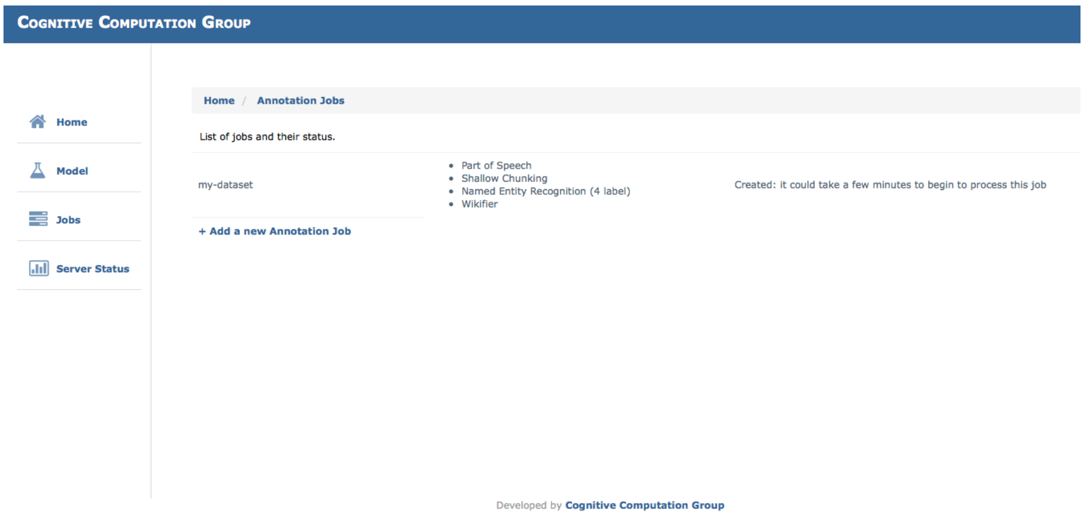

Cognitive Computation Group
Home
Model
Jobs
Server Status
Welcome to Illinois CloudNLP!
You can now:
Train Models
Annotate a Set of Documents (create a Job)
Calculate estimated cost
Developed by Cognitive Computation Group

Figure 2: Main selection pane in ILLINOISCLOUDNLP interface


Cognitive Computation Group
Home
Model
Jobs
Server Status
Home / Annotation Jobs
+ Add a new Annotation Job
Developed by Cognitive Computation Group

List of jobs and their status.

my-dataset	• Part of Speech • Shallow Chunking • Named Entity Recognition (4 label) • Wikifier	Created: it could take a few minutes to begin to process this job

Figure 4: ILLINOISCLOUDNLP pane showing jobs in progress

directory, and name the two directories with the label they would like the classifier to assign to that type of document. They put these two directories into a single root directory, and create a zip from that directory.

After selecting the option *"Train Models"* from the main menu, the user will be prompted to upload the zip file (see Figure 5). Clicking the *"Train it"* button uploads the data to an S3 bucket and launches a single Amazon EC2 node that runs the training process. Once completed, the interface alerts the user, and a new model with the name provided by the user appears in the list of available models in the *Model* pane. It also appears as an available annotation component when starting an annotation task (see Figure 3).

5.5. Viewing and Downloading the Annotated Data

Once the documents in the collection have been processed, the annotations are stored using Amazon S3 in the data bucket the user named when the task was specified. The processed documents can be viewed in a concise, human-readable form using the ILLINOISCLOUDNLP client's visualization interface (see Figure 6). Data labeled with the ILLINOISCLASSIFIER component appear as views with single labels.

By default, these annotations will remain on the Amazon S3 cluster storage service for later use (for a use case that illustrates why this can be helpful, see Section 6.). They can be downloaded from Amazon S3 using a third party

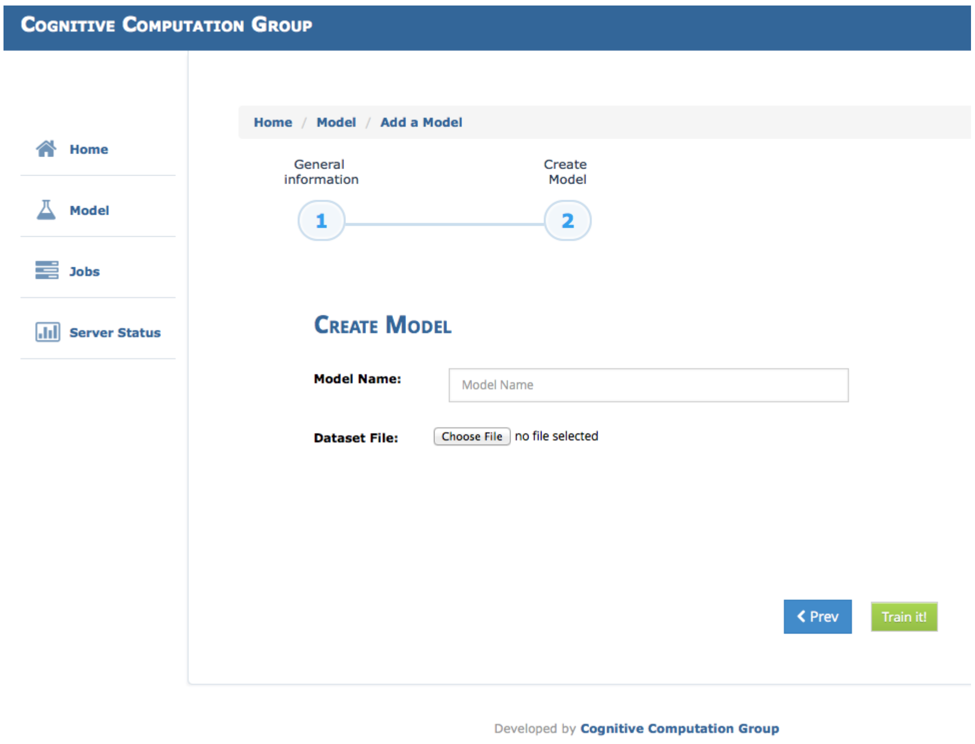


Figure 5: ILLINOISCLOUDNLP pane for setting up a classifier training task

application such as S3cmd[3] for local use, and deleted from the S3 storage if desired.

6. An Example Application

For a recent project, we wanted to annotate 3.4 million documents from TAC KBP 2012 and 2013 (Ellis et al., 2012; Ellis et al., 2013) with a set of our NLP tools. A server with 128G RAM and two 6-Core, 2.4GHz Intel Xeon E5645 processors would have taken more than two weeks to process this collection. In the past, the best we could do was to split our dataset and manually distribute it to several servers running the curator. However, this is limited by the number of available servers and the need to share them between multiple projects. With ILLINOISCLOUDNLP, we were able to annotate 3.05 million documents with the NLPCURATOR's Wikifier, NER, POS and Chunker components, in only 20 hours and for $500. In addition, the outcome is a set of serialized NLPCURATOR Record data structures (Clarke et al., 2012) stored on Amazon S3; these can be easily deserialized and used in a number of programming languages. But working on the Amazon EC2 cloud and using its S3 storage has other advantages too. In almost any project involving a large set of documents, it is likely that there is a need for some batch processing to generate useful statistics from the annotated dataset, such as, for example, building inverted indexes on words or entity mentions. Often this involves a lot of time and energy, not only in building and maintaining further infrastructure to support these operations, but also in running them on slow local hardware.

However, having all of our annotated documents in Amazon S3, we are able to take advantage of other Amazon services, specifically AWS Elastic MapReduce (EMR). Using EMR, we can easily run MapReduce jobs on those annotated documents using straightforward MapReduce code. Through a very simple setup, EMR allows us to calculate a wide range of interesting statistics on the text we just processed. For example, counting frequency and co-occurrence of entities, or answering more sophisticated questions like what is the most popular adjective to describe a noun such as "president".

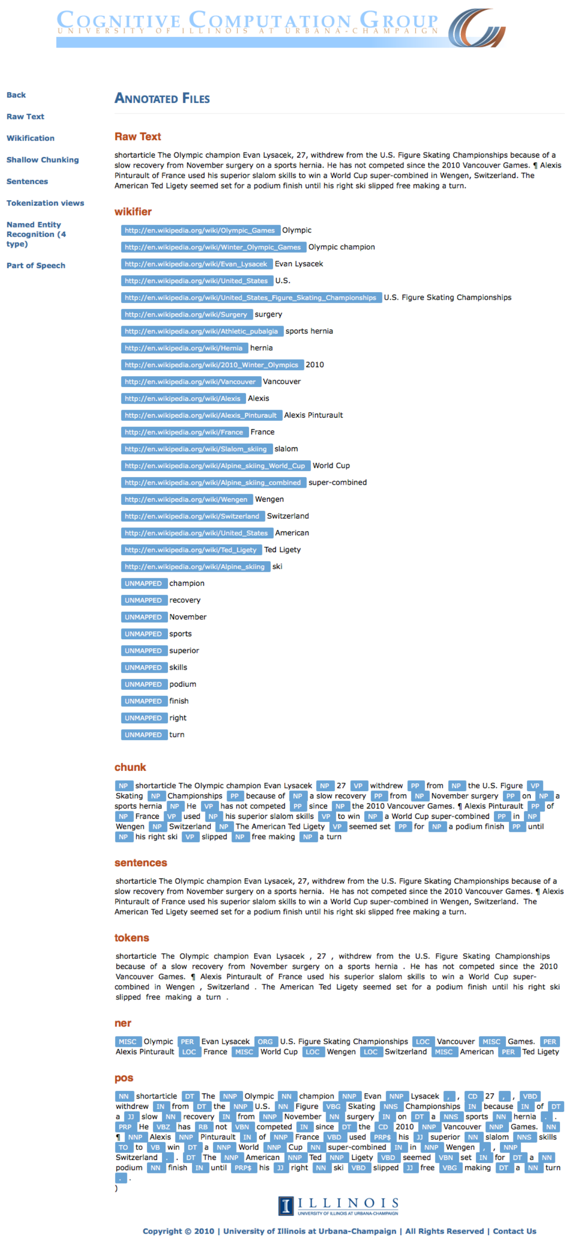
7. Related Work

We discuss below a number of other machine learning/NLP frameworks built around cloud services, with varying objectives and expected user expertise.

7.1. API-based Services

A common way to provide cloud-based NLP services is through an API. Two such services are AlchemyAPI (2014) and Mashape (2014). AlchemyAPI is an API dedicated to text processing. Mashape is an open-domain API marketplace which hosts several text analytics APIs, one example being AYLIEN (2014). By virtue of being an API, these services require more user expertise and, in addition, both services are significantly more expensive than ILLINOISCLOUDNLP. For example, the Mashape AYLIEN API with minimal functionality costs $200 per month and only allows 6000 transactions per day. It would take 500 days to annotate our 3 million documents, at a cost of over $3000.

[3] `http://s3tools.org/s3cmd`


COGNITIVE COMPUTATION GROUP
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Back
Raw Text
Wikification
Shallow Chunking
Sentences
Tokenization views
Named Entity Recognition (4 type)
Part of Speech

ANNOTATED FILES

Raw Text

shortarticle The Olympic champion Evan Lysacek, 27, withdrew from the U.S. Figure Skating Championships because of a slow recovery from November surgery on a sports hernia. He has not competed since the 2010 Vancouver Games. ¶ Alexis Pinturault of France used his superior slalom skills to win a World Cup super-combined in Wengen, Switzerland. The American Ted Ligety seemed set for a podium finish until his right ski slipped free making a turn.

wikifier

http://en.wikipedia.org/wiki/Olympic_Games Olympic
http://en.wikipedia.org/wiki/Winter_Olympic_Games Olympic champion
http://en.wikipedia.org/wiki/Evan_Lysacek Evan Lysacek
http://en.wikipedia.org/wiki/United_States U.S.
http://en.wikipedia.org/wiki/United_States_Figure_Skating_Championships U.S. Figure Skating Championships
http://en.wikipedia.org/wiki/Surgery surgery
http://en.wikipedia.org/wiki/Athletic_pubalgia sports hernia
http://en.wikipedia.org/wiki/Hernia hernia
http://en.wikipedia.org/wiki/2010_Winter_Olympics 2010
http://en.wikipedia.org/wiki/Vancouver Vancouver
http://en.wikipedia.org/wiki/Alexis Alexis
http://en.wikipedia.org/wiki/Alexis_Pinturault Alexis Pinturault
http://en.wikipedia.org/wiki/France France
http://en.wikipedia.org/wiki/Slalom_skiing slalom
http://en.wikipedia.org/wiki/Alpine_skiing_World_Cup World Cup
http://en.wikipedia.org/wiki/Alpine_skiing_combined super-combined
http://en.wikipedia.org/wiki/Wengen Wengen
http://en.wikipedia.org/wiki/Switzerland Switzerland
http://en.wikipedia.org/wiki/United_States American
http://en.wikipedia.org/wiki/Ted_Ligety Ted Ligety
http://en.wikipedia.org/wiki/Alpine_skiing ski
UNMAPPED champion
UNMAPPED recovery
UNMAPPED November
UNMAPPED sports
UNMAPPED superior
UNMAPPED skills
UNMAPPED podium
UNMAPPED finish
UNMAPPED right
UNMAPPED turn

chunk

NP shortarticle The Olympic champion Evan Lysacek NP 27 VP withdrew PP from NP the U.S. Figure VP Skating NP Championships PP because of NP a slow recovery PP from NP November surgery PP on NP a sports hernia NP He VP has not competed PP since NP the 2010 Vancouver Games. ¶ Alexis Pinturault PP of NP France VP used NP his superior slalom skills VP to win NP a World Cup super-combined PP in NP Wengen NP Switzerland NP The American Ted Ligety VP seemed set PP for NP a podium finish PP until NP his right ski VP slipped NP free making NP a turn

sentences

shortarticle The Olympic champion Evan Lysacek, 27, withdrew from the U.S. Figure Skating Championships because of a slow recovery from November surgery on a sports hernia. He has not competed since the 2010 Vancouver Games. ¶ Alexis Pinturault of France used his superior slalom skills to win a World Cup super-combined in Wengen, Switzerland. The American Ted Ligety seemed set for a podium finish until his right ski slipped free making a turn.

tokens

shortarticle The Olympic champion Evan Lysacek , 27 , withdrew from the U.S. Figure Skating Championships because of a slow recovery from November surgery on a sports hernia . He has not competed since the 2010 Vancouver Games. ¶ Alexis Pinturault of France used his superior slalom skills to win a World Cup super-combined in Wengen , Switzerland . The American Ted Ligety seemed set for a podium finish until his right ski slipped free making a turn .

ner

MISC Olympic PER Evan Lysacek ORG U.S. Figure Skating Championships LOC Vancouver MISC Games. PER Alexis Pinturault LOC France MISC World Cup LOC Wengen LOC Switzerland MISC American PER Ted Ligety

pos

NN shortarticle DT The NNP Olympic NN champion NNP Evan NNP Lysacek , , CD 27 , , VBD withdrew IN from DT the NNP U.S. NN Figure VBG Skating NNS Championships IN because IN of DT a JJ slow NN recovery IN from NNP November NN surgery IN on DT a NNS sports NN hernia . . PRP He VBZ has RB not VBN competed IN since DT the CD 2010 NNP Vancouver NNP Games. NN ¶ NNP Alexis NNP Pinturault IN of NNP France VBD used PRP$ his JJ superior NN slalom NNS skills TO to VB win DT a NNP World NNP Cup NN super-combined IN in NNP Wengen , , NNP Switzerland . . DT The NNP American NNP Ted NNP Ligety VBD seemed VBN set IN for DT a NN podium NN finish IN until PRP$ his JJ right NN ski VBD slipped JJ free VBG making DT a NN turn . .
)

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Copyright © 2010 | University of Illinois at Urbana-Champaign | All Rights Reserved | Contact Us

Figure 6: Viewing annotated data.

Semantria (2014) provides a variety of NLP services based on the Lexalytics Salience engine, but is accessed only through an API, or through Microsoft Excel. The baseline cost is $450 per month for academic users ($900 for commercial users), and the number of transactions is limited to 100,000 (where each transaction represents a discrete piece of text to be annotated). Again, it requires more expertise and is more expensive than ILLINOISCLOUDNLP.

AnnoMarket (Tablan et al., 2013) is based on GATE tools (Cunningham et al., 2002) and provides APIs for text processing, but also allows a user to create their own API and sell it to others. Unlike the services above, the cost is hourly based on machine usage. At present, AnnoMarket has good multilingual support for some basic NLP tasks but limited diversity in types of NLP tools. The key difference between these services and ILLINOISCLOUDNLP is that simply by virtue of being APIs, they require significant user expertise for even simple tasks.

7.2. NLP Processing Services

GATECLOUD (2014) is a cloud-based version of the GATE suite of NLP tools, and it offers a number of specialty services. The relevant functionalities are similar to those offered by ILLINOISCLOUDNLP, but only include annotations out-of-the-box for a limited number of tools – named entities, measurements, and numbers. Charges are calculated by the hour.

7.3. Machine Learning Cloud Services

There are other cloud computing services that support machine learning and/or algorithm development. GoogleAPI (2014) provides a service for learning and using a classifier, but does not support the range of Text Analytics tools that we do. Moreover, users must send their data to Google, and build their classifier using its API. The free quota of predictions is limited; going beyond 40k queries/day requires direct negotiation with Google. GraphLab (2014) provides the infrastructure for algorithms development on the cloud, but targets algorithm developers rather than application programmers as we do. Moreover, it does not have an existing model for text analytic components as we do.

7.4. Summary

It is important to note that for all of the services described in this section, the issues of of data privacy and security are not handled in a clear and transparent way. For example, it is sometimes unclear who provides the cloud computing resources, and a third party is involved in handling billing and in managing access to computing resources. Many of these services require an ongoing financial commitment, limit the amount of processing per day, or offer only relatively low-complexity NLP analytics.

On the other hand, ILLINOISCLOUDNLP provides a state-of-the-art suite of tools for English text analytics via a very simple interface. Our analytics tools have all been individually described and evaluated in peer-reviewed academic publications. There is no third party billing agent, and users maintain direct control over their data, which is processed and stored in a user-owned account on Amazon EC2/S3 machines and protected via their Amazon credentials. Rather than paying a monthly subscription for a fixed upper limit on the number of documents that can be processed, users pay as they go and process what they need, when they need to.

8. Conclusions and Future Work

ILLINOISCLOUDNLP is a cloud-based service that allows users to process plain text documents with a suite of state-of-the-art NLP tools via a simple user interface, and to train text classifiers using a generic feature representation. This solution is cost-effective for end users with intermittent processing needs and requires no NLP, Machine Learning, or Cloud Computing expertise. As a demonstration, we have developed a simple application over ILLINOISCLOUDNLP by processing 3.05 million documents from the TAC KBP tasks with segmentation, Part of Speech, Named Entity Recognition, and Wikification in approximately 20 hours, at a cost of approximately US$500. This task would require about a month of continuous processing on a single local server, a non-trivial installation effort, and a lot of human expertise and supervision in case of failure.

The ILLINOISCLOUDNLP initial release is a working prototype: it should already be a useful resource in its own right. However, we plan to improve its capabilities along two distinct dimensions. Additional Text Analytics components will be added to ILLINOISCLOUDNLP based on other stand-alone NLP components developed by the Cognitive Computation Group[4]. The ILLINOISCLASSIFIER component will be extended to use features extracted from NLPCURATOR annotations to allow more expressive models to be learned. In addition, the learning framework will be extended to allow chunk-level annotations such as Named Entities to be learned.

We also plan to investigate caching mechanisms that go beyond the current single-user model, and allow voluntary sharing of document annotations.

ILLINOISCLOUDNLP can be downloaded from `http://cogcomp.cs.illinois.edu/software/IllinoisCloudNLP` under an Academic Use license. Its home page contains a more detailed overview of its use, together with pointers to useful third party resources.

9. Acknowledgements

This research was supported by: the Multimodal Information Access & Synthesis Center at UIUC, part of CCICADA, a DHS Science and Technology Center of Excellence; the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053; DARPA, under agreement number FA8750-13-2-0008; and NSF grant #SMA 12-09359. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the aforementioned organizations.

10. References

AlchemyAPI. (2014). http://www.alchemyapi.com/. Accessed: March 2014.

Amazon.com. (2014). Amazon web services. http://aws.amazon.com/. Accessed: March 2014.

AYLIEN. (2014). AYLIEN intelligence. http://aylien.com/. Accessed: March 2014.

Cheng, X. and Roth, D. (2013). Relational inference for wikification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Clarke, J., Srikumar, V., Sammons, M., and Roth, D. (2012). An NLP Curator (or: How I learned to stop worrying and love NLP pipelines). In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*, 5.

Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *ACL*.

Ellis, J., Li, X., Griffitt, K., Strassel, S. M., and Wright, J. (2012). Linguistic resources for 2012 knowledge base population evaluation. In *Text Analysis Conference (TAC)*.

Ellis, J., Getman, J., Mott, J., Li, X., Griffitt, K., Strassel, S. M., and Wright, J. (2013). Linguistic resources for 2013 knowledge base population evaluations. In *Text Analysis Conference (TAC)*.

GATECLOUD. (2014). GATECLOUD: Text solutions on the cloud. https://gatecloud.net/. Accessed: March 2014.

GoogleAPI. (2014). Google Prediction API. https://developers.google.com/prediction/. Accessed: March 2014.

GraphLab. (2014). The GraphLab Project. http://graphlab.org/projects/index.html. Accessed: March 2014.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). Ontonotes: The 90% solution. In *HLT-NAACL*.

Kearns, M. J. and Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

Mashape. (2014). Mashape, the cloud api platform. https://www.mashape.com. Accessed: March 2014.

Punyakanok, V. and Roth, D. (2001). The use of classifiers in sequential inference. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 995–1001. MIT Press.

Ratinov, L. and Roth, D. (2009). Design challenges and misconceptions in named entity recognition. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, 6.

Rizzolo, N. and Roth, D. (2010). Learning based Java for rapid development of NLP systems. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta, 5.

Roth, D. and Zelenko, D. (1998). Part of speech tagging using a network of linear separators. In *Coling-Acl, The 17th International Conference on Computational Linguistics*, pages 1136–1142.

Semantria. (2014). Semantria: Text analytics and sentiment analysis. https://semantria.com/. Accessed: March 2014.

Tablan, V., Bontcheva, K., Roberts, I., Cunningham, H., and Dimitrov, M. (2013). Annomarket: An open cloud platform for nlp. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Sofia, Bulgaria.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

[4]For examples of these tools, please see `http://cogcomp.cs.illinois.edu/software` and `http://cogcomp.cs.illinois.edu/demos`.