Automatic classification of work-related injury and illness narratives

Organisation: U.S. Bureau of Labor Statistics

Author(s): Alexander Measure

Date: 2/26/2020

Version: 2.1

1. Background and why and how this study was initiated

(as many sentences as necessary, as few as possible)

U.S. laws require many organizations to keep records of the circumstances of work related injury and illness. Each year the Bureau of Labor Statistics (BLS) collects about 300,000 of these through our Survey of Occupational Injuries and Illnesses (SOII). We then read and categorize this data to provide information about the number, rate, and circumstances of injuries and illnesses experienced by American workers. Among other items, we identify occupation of the worker based on their job title, and the nature of the injury, part of body injured, event, and sources of injury from the injury narrative. For most of our history this work was done by hand, but this is not a perfect solution. Manual coding requires significant labor, in our case estimated at about 25,000 hours per year. To perform this quickly we must therefore use a large number of people, and this requires that we train and monitor these individuals to ensure they classify injuries in a consistent manner. This is far from easy. In one experiment, for example, we found that when we asked different experts to independently code the same cases, any two experts only chose the same codes for the same cases about 70% of the time.

This project was also influenced by a number of other factors, including the following:

- In 2009, researchers from Purdue University and Liberty Mutual published research suggesting that ML techniques might be very effective for injury classification (see "Bayesian methods: a useful tool for classifying injury narratives into cause groups" by Lehto, Wellman, and Corns).

- At around the same time, BLS received additional funding to study weaknesses in the Survey of Occupational Injuries and Illnesses. This research revealed that categorization errors were one source of problems.
- Recently launched free online courses on machine learning and natural language processing made it much easier for staff to quickly learn about the latest machine learning and natural language processing techniques.

Ultimately these factors led BLS management to allocate resources to the automated categorization of injuries and illnesses, including creating a new position to spearhead this effort.

One of our first tasks was to evaluate potential solutions. We initially considered 3 options:

1. A rule-based system developed by a popular statistical software company.
2. A machine learning system developed by academics.
3. A machine learning system, developed in-house using the free and open-source library scikit-learn.

We launched pilot projects to evaluate each of these and ultimately determined that open source software provided the best quality coding and the lowest cost for our task.

2. Data

2.1 Input Data **(short description)**

The primary inputs to our task are digital text narratives describing the occupations of injured workers and the various circumstances of their injuries and illnesses. Although there is significant variation in the length and content of these narratives, a typical narrative might resemble the following:

Narrative	**Codes assigned by BLS**
Job title: RN **What was the worker doing?** Helping patient get into wheel chair **What happened?**	Occupation: 29-1141 (registered nurse) Nature: 1233 (strain) Part: 322 (lower back) Event: 7143 (overexertion in catching)

Patient slipped and employee tried to catch her **What was the injury or illness?** Strained lower back **What was the source?** Patient	Source: 574 (patient) Secondary source: None

Company and industry specific terminology are common, as are spelling errors. Additional inputs include the industry classification of the worker's employer, the employer's name, and a 12 item checkbox indicating the general focus of the worker's occupation. The training targets are the occupation and injury and illness codes that were assigned by human coders. Each case receives six codes indicating the occupation of the worker, the nature of their injury or illness, the part of body affected, the event that caused the incident, and the source and secondary source of the injury or illness. The occupation is assigned according to the Standard Occupation Classification (SOC) system. The remaining codes are assigned according to version 2.01 of the Occupational Injury and Illness Classification System (OIICS). When our autocoding research began our training and evaluation data consisted of approximately 261,000 records, but it later grew to more than 2 million and continues to grow each year.

2.2 Data Preparation

(e. g.: Data Cleaning, Normalisation… – or: none)

Although we initially conducted a variety of data cleaning, preparation, and normalization experiments, we ultimately found insufficient benefit to justify the added complexity. Our best autocoders do no stop-word removal, stemming, or other extensive preprocessing beyond tokenization using simple heuristics (i.e. separate words on blank spaces).

2.3 Feature Selection

(yes/no, if yes: how, why)

We did very little feature selection. We focused primarily on exploring inputs that humans used when completing similar tasks.

2.4 Output data

(short description)

Six classifications are assigned to each case based on the SOC and OIICS 2.01 classification systems:

1. Occupation (according to the SOC classification system)
2. Nature of injury (OIICS)
3. Part of body injured (OIICS)
4. Event that caused injury (OIICS)
5. Source of injury (OIICS)
6. Secondary source of injury (OIICS)

We eventually trained ML models to assign codes to each of these. Each model calculates the probability that each possible code is correct. The highest probability code is automatically assigned if it exceeds a predetermined probability threshold.

3. Machine Learning Solution

3.1 Models tried

(e. g.: Multi-Layer-Perceptron, Random Forest, SVM, ...)

We tried many popular approaches to text classification including Logistic Regression, Support Vector Machines, variants of Naïve Bayes, Random Forests, multilayer perceptrons, and many variations of convolutional and recurrent neural networks. The results of some of these experiments are described in 2 papers:

- Automated Coding of Worker Injury Narratives (https://www.bls.gov/osmr/research-papers/2014/pdf/st140040.pdf)
- Deep neural networks for worker injury autocoding (https://www.bls.gov/iif/deep-neural-networks.pdf)

3.2 Model(s) finally selected and the criterion

(i.e.: which model was why seen being the best?)

How was the final model selected? How did it meet all or most of the objectives or expected added value of the study?

The primary purpose of our project was to improve the overall quality of coding, which at the time was done entirely by humans. To solve this we needed to measure and compare both human and automated coding quality. We did this by selecting a representative sample of cases that had already gone through our manual coding process. We then hid the codes that had been assigned to each of these and asked a panel of coding experts to recode each case from scratch. These expert codes formed our “gold standard” against which we evaluated the quality of both our existing (human-based) and automated coding options. Our evaluations showed that our automated coders (which were not trained on any of the cases in the gold standard data) were more accurate than our manual process when coding the entire gold standard dataset. Although this indicated that automated approaches were promising, it did not tell us the optimal combination of manual and automated coding.

Since our autocoders not only predict codes but also calculate probabilities associated with these predictions, and since these probabilities are empirically closely related to the actual probability that the codes are correct, we decided to use the predicted probability to control the amount of autocoding. In the simplest case this is done by selecting a probability threshold and only allowing autocoding when the predicted probably exceeds that threshold. To find the best threshold we used simulation on the gold standard. Specifically, for each possible threshold between 0 and 1 in increments of .01:

- We automatically assigned codes to the gold standard using machine learning if the associated probability exceeded the chosen threshold.
- We assumed all other codes would be assigned by human staff and would therefore match the codes that had been manually assigned to these cases.
- We then calculated the overall accuracy and macro F1 score of the combination of manual and computer coding

For each task we selected the probability threshold that produced the highest overall macro-F1-score for the combination of manual and computer coding, mitigated initially only by a competing desire to increase the quantity of autocoding slowly so other processes had sufficient time to adapt.

3.3 Hardware used

(e.g.: Intel Core i5-6300U, 2.4GHz)

We initially used standard laptops with 2-4 cores and 8-16 gigabytes of RAM (RAM being the main limiting factor). When we switched to neural networks we used 4 Titan X Pascal GPUs, each with 12 gigabytes of RAM and 3,584 cores.

3.4 Runtime to train the model

(e.g.: 2 hours for 500,000 training samples and 25 features)

This depended on the model and the amount of training data available at that time. Some algorithms, like naïve Bayes, trained very fast (nearly instantly on small datasets). Others, like some of the deep neural networks, could take several days or even weeks to train on millions of records.

4. Results

(e. g. in terms of RMSE, MAE, distributional accuracy [*], F1 (micro or macro), recall, accuracy, (threshold,) ..., perhaps as a table for different situations (if available))

[*]: If used: How did you measure distributional accuracy? By proportions, moments, quantiles, correlations, …?

See our papers for results and descriptions of the metrics calculations. Our primary metrics were accuracy and macro F1 score, calculated using scikit-learn.

- Automated Coding of Worker Injury Narratives
- Deep neural networks for worker injury autocoding

5. Code/programming language

(e.g. the Python code is stored in GitHub)

All autocoders were created using the Python programming language. The neural network autocoders were created using the free and open-source Tensorflow and Keras libraries. All others were created using the free and open-source scikit-learn library. All evaluation metrics were calculated using the scikit-learn accuracy_score and f1_score functions. The code for the neural network autocoder is available on GitHub at https://github.com/USDepartmentofLabor/soii_neural_autocoder. The logistic regression autocoders are very similar to the autocoders described in this autocoding tutorial. See our papers for additional details.

6. Evolution of this study inside the organisation

(e. g.: Collaboration within the organisation? Has this study advanced ML within the organisation?)

Many stakeholders from all major branches of the organization were consulted, including information technology, methodology, and subject matter. Eventually a cross-functional team was created to oversee the development of the autocoder. Most of the work is still done by a small number of individuals however because of a shortage of staff familiar with Python programming and neural networks.

When the project started there were no other groups using machine learning for text classification in BLS. The success of our project however has inspired many similar projects and a number of efforts to use these techniques more broadly, including formal and informal meetings with programs involved in similar projects, dozens of internal presentations, and internal trainings and training materials. We continue to advise and collaborate with a number of related projects on a frequent basis.

7. Is it a proof of concept or is it already used in production?

(If it is a proof of concept: Was it successful? How will its results prospectively be used in the future?)

This project is in production and has been in production since 2012, when it was initially used to assist with data review. Rollout was a long and gradual process that started small and gradually grew over many years. Today more than 85% of codes are assigned by a neural network.

There were many challenges along the way but some of the biggest include the following:

1. Lack of relevant skills and knowledge: BLS has lots of survey statisticians, economists, database administrators, web developers and SAS and Java programmers, but natural language processing, machine learning, and Python are not things that any of

these people are normally trained in and these skills are critical for implementing modern machine learning systems.

2. Organizational structure: Effective machine learning projects require a combination of product, methodology, and information technology expertise, but the organizational structure separates these skills into different divisions reporting to different leaders and this makes it difficult to align these skills toward common goals. Ultimately, intervention from senior management was required to align these activities.

7.1 What is now doable which was not doable before?

Our machine learning autocoders allow us to categorize occupation and injury and illness data more accurately than trained human staff. This improves the quality of our estimates and frees up staff to spend more time on other important activities like data review and additional data collection.

7.2 Is there already a roadmap/service journey available how to implement this?

(as many sentences as necessary, as few as possible)

The system is already implemented. For the first couple of years the system was only used to help with review of manually assigned codes. Automated code assignment started in 2014 and initially accounted for a tiny amount of coding. This gradually expanded over time as people became more comfortable with it.

7.3Who are the stakeholders?

The people most directly affected are our human coders. As the computers have automated more coding, their role has shifted from lots of manual coding to more review and data collection. Although we initially expected some pushback, our experience has mostly been that the coders were happy to have assistance. The other major stakeholders are our data users. We presented our work in a wide variety of public venues and to a variety of advisory committees to gather their input.

7.4 Robustness

We have a variety of processes to ensure the autocoder works as desired. Before any model is put into production it is evaluated against a representative sample of "gold standard" cases. The model is only used if it shows better performance than the previous automated or human based approaches. After the model is placed into production we use the following processes to verify that it continues to work correctly:

1) Manual review: We ask the human staff that used to do all coding to manually review every automatically assigned code and change it if it is incorrect. We also ask them to notify us if they notice any systematic problems.
2) Hold back review: We randomly select a sample of cases to exclude from automated coding. We do this to see how our human staff code cases without the assistance of a computer. This allows us to later update our gold standard measurements of human quality. It also helps us identify issues in automated coding by allowing us to compare the human assigned codes to codes the computer would have assigned.
3) Additional levels of review: As we did with purely manual coding, we conduct a variety of additional levels of coding review primarily in the regional and national offices. This includes targeted review of known and suspected problem areas.
4) Key metrics review: We also periodically check a variety of metrics to verify that the autocoder is working as expected. For example, we look at how often human coders change computer assigned codes. We are in the process of using this information to train another machine learning model to automatically identify codes that need additional review.

7.5 Fall Back

The human staff that used to do all of the coding still work at BLS, and still assign codes to cases that the autocoder does not. If the autocoder started making lots of errors they would likely notice and notify us. If necessary we could shut down the autocoder and return to full manual coding, although this would reduce the amount of time our human staff could spend on other important activities like data collection and review.

8. Conclusions and lessons learned

(e.g.: ML can be used for editing but one has to have the following points in mind ...)

This project has greatly advanced the knowledge and awareness of machine learning and natural language processing techniques in my organization.

9. Potential organisation risk if ML solution not implemented

(as many sentences as necessary, as few as possible)

In our case, the main risk was lower quality data. Our evaluations showed that automated coding produced higher quality coding.

10. Has there been collaboration with other NSIs, universities, etc?

(yes/no, if yes: which ones?)

Yes. In the early days the project received assistance from autocoding researchers at Purdue and Liberty Mutual. We were later introduced to other techniques through the online courses of Andrew Ng (Machine Learning), Michael Collins (Natural Language Processing), Chris Manning and Dan Jurafsky (Natural Language Processing), and Geoffrey Hinton (Neural Networks). We also relied heavily on a wide variety of research in machine learning and text classification conducted by many researchers from all over the world (see the references in our papers for some of these).

Next Steps

(as many sentences as necessary, as few as possible)

We plan to continue to use machine learning for Survey of Occupational Injury and Illness coding and we also plan to expand the use of machine learning to other coding tasks and data review and record matching projects.