Center for Data Science and Public Policy

THE UNIVERSITY OF CHICAGO

Skills-ML: An Open Source Python Library for Developing and Analyzing Skills and Competencies from Unstructured Text

Authors

Tristan Crockett, Eddie Lin, Matt Gee, Christina Sung

November 5, 2018

Table of Contents

Abstract

This paper describes Skills-ML, an open source Python software library for applying natural language processing and machine learning algorithms to labor market problems such as automation. Skills-ML allows the user to take unstructured and semistructured text, such as job postings, and perform relevant tasks such as competency extraction and occupation classification using existing or custom competency ontologies. Skills-ML is designed to have no infrastructural dependencies, allowing the user to easily run these tasks on their laptop or an available computing cluster. Skills-ML has been used to produce large open research datasets from a raw job posting dataset, without needing to spend a long time transforming data and interfacing with generic ML and NLP libraries. This paper describes the capabilities and general interface of Skills-ML and includes descriptive statistics for applying the library to a job posting sample. Skills-ML is available at https://www.github.com/workforce-data-initiative/skills-ml

Introduction

Artificial Intelligence (AI), machine learning, and large-scale data analytic technologies are being used to automate routine tasks, improve decision-making, and redesign the customer experience in a wide variety of business processes in both the public and private sectors. The relatively recent dramatic increase in both the algorithmic performance and public awareness of AI has fueled both an active public debate, as well as a pressing research agenda among academic economists and public policy researchers about the future of work. Central to this debate is the need to better understand how automation increases the value of some skills while making others partially or entirely obsolete.

Autor, Levy, and Murnane (2003) set the stage for assessing the exposure of economic activities to the emergence of information technology, but no such framework exists for AI, which goes beyond merely the execution of instructions. Advances in cloud computing, computing power, and data availability have allowed deep learning algorithms to make remarkable progress in executing tasks under ambiguity. However, measurement of these advances has been challenging. Acemoglu and Restrepo (2018) look at the expansion of industrial robots, but these measures are tracked by industry and year and primarily representative of manufacturing industries. Finally, and perhaps most important, Acemoglu and Restrepo (2018) state how wages and the incentives for automation and the creation of new tasks respond to policies, factor prices, and supplies. Frey and Osborne (2017) look at all occupations, but their assessment of exposure is highly subjective.

At the same time, educational institutions and employers are increasingly instrumenting their education, training, and hiring processing and using data to change the way that labor supply and demand line up. On the demand side, employers and their HR vendors are using these technologies to revamp the end-to-end talent sourcing process. These technologies are increasingly being used in performance analytics to determine which skills, educational, and career backgrounds are most important in job performance and how this information can be used to predict which job candidates would be good hires. These technologies are also being

used to improve outreach and recruitment in the employer's talent network and improve decision-making and connections in the talent screening and hiring process. On the supply side, universities and colleges are exploring how to use these technologies to improve connections with current and future students and improve outreach, enrollment, education, and student services. Governments are also using these technologies to improve services.

A common thread across each of these needs, as well as the needs of other labor market research areas such as policy changes, is the ability to identify, relate, and translate competencies. This is what Skills-ML attempts to address.

Applications of Skills-ML

Skills-ML can be used to build applications for many uses in diverse research areas and industries helping many use cases across different audience types. Several examples are detailed below.

Underemployed Job Seeker
This type of job seeker may be helped by an application that can take his current occupation, current location, and a dream job to build a "roadmap" to that dream job. An application developer can use Skills-ML to classify occupations and extract competencies from local job postings. With this information, the application can find the competencies that the job seeker should acquire that are not covered by his current occupation, and connect him with local opportunities to gain those competencies.

Military Spouse
Military spouses often suffer from geographical skills mismatch, in which they have lived in many different places and held different jobs in each of those places. Similar job titles may consist of different competencies depending on location so when looking for jobs in a new place, the spouse may not know what to search for. An application that could help them is an occupation suggestion engine based on their resume data and local job postings from the locations that they have worked in. Similar to the underemployed job seeker use case, local job postings from those places would be used to classify occupations and extract competencies to build a competency profile. Where this differs from the underemployed job seeker use case is that instead of targeting a specific occupation in the new location, it would compare all occupations in the new location in competency space to the user's competency profile to suggest occupations that most closely match their current competency profile.

Community College
To succeed, community colleges and other training providers would like to keep their course offerings relevant from both a time and geographic standpoint. They want to know which occupations and competencies will likely be relevant in their area on an intermediate to long-term scale (e.g. a year). They can feed both the local job postings and their own course syllabi into Skills-ML to extract competencies from each to understand the overlap in competencies, which will identify what is relevant in their courses. They can use this information to modify their curriculum by creating or modifying courses for in-demand competencies or cancel courses for competencies found to be irrelevant.

Economics Research
Economists may want to understand the production function for hiring. They can use Skills-ML to classify occupations and extract competencies from nationwide job postings to analyze firm recruitment efforts alongside other economic data to estimate the production function.

Tasks Available in Skills-ML

The applications described above feature a large amount of overlap in terms of the tasks that are necessary to complete them.

Occupation Classification

The process of taking a document (such as a job posting) and inferring the occupation described by the document is common. A lot of external data (e.g. from labor surveys) is collected at the occupation level, so being able to classify an occupation from a document enables a lot of analyses.

Without Skills-ML, assembling an occupation classification pipeline involves writing a significant amount of “glue code” (code that transforms data between formats needed by different off-the-shelf libraries) to interface with:

- NLP library like NLTK for parsing text
- Embedding library like word2vec for vectorizing text
- ML library like scikit-learn for training and evaluating models

With Skills-ML, a user can import a pipeline that works with JSON documents as input and handles all of the vectorization, training and testing tasks. It allows the user to define groups of occupations as the unit of evaluation without writing extra code. In addition, the pipeline enforces the use of metadata describing each phase which helps the user manage the results of their experimentation.

Competency Extraction

The process of taking document text and finding the competencies described within it is important to many analyses. Competencies are a common component in many documents

related to the labor market (job postings, course descriptions, resumes) and extracting competencies allows us to compare these documents in a common language.

There are a variety of methods to extract competencies, each with their own positives and negatives. Some are simple NLP methods which would require working directly with an NLP library without Skills-ML. Others are based on matching words or phrases from a known list of competencies, which would require acquiring and formatting the known competencies in a specific way, which is not necessary using Skills-ML's prebuilt ontologies. By implementing a variety of these methods using the same common input and output formats, Skills-ML allows the user to try different methods and evaluate them without writing glue code.

In addition, since both Occupation Classification and Competency Extraction use the same document input format, analyses that require both tasks are much easier to get off the ground using Skills-ML.

Skills-ML Concept Overview

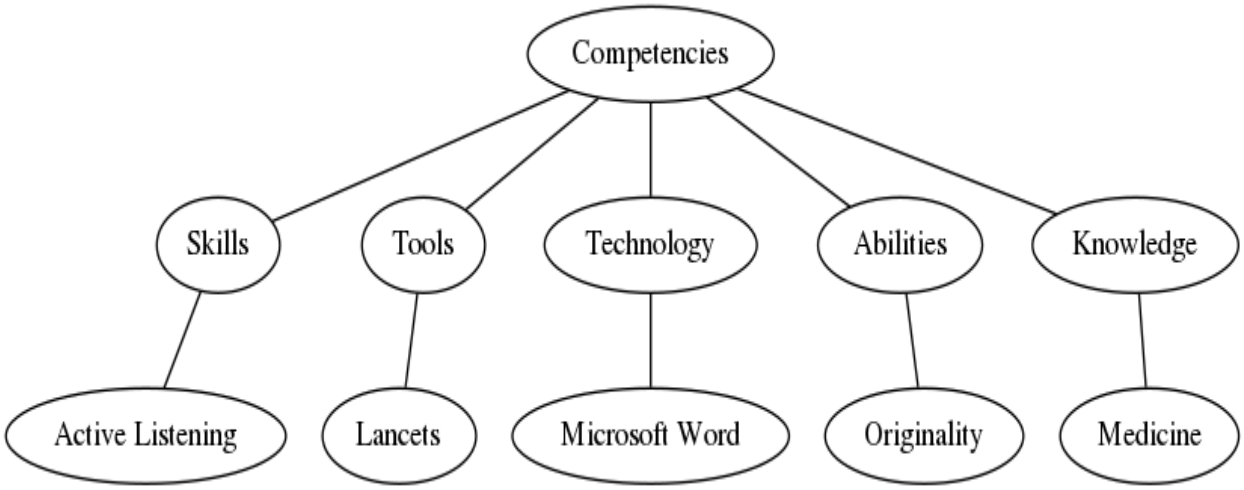
This section describes a few general concepts used by the Skills-ML library, including how they are used outside of Skills-ML, as well as what they mean within the library. These concepts are referenced heavily in the rest of the paper. Terms formatted with Italics are used throughout this section to denote specific classes or functions within Skills-ML.

Skills and Competencies

The terms 'skill' and 'competency' are used interchangeably in many contexts, and they are similar but differ in important ways.

A competency is any expertise or talent that is useful for a job. Examples of competencies can vary widely but include developed capacities (e.g. active listening), proficiency with tools or technology (e.g. lancets, Microsoft Word), innate abilities (e.g. originality), and academic knowledge (e.g. medicine).

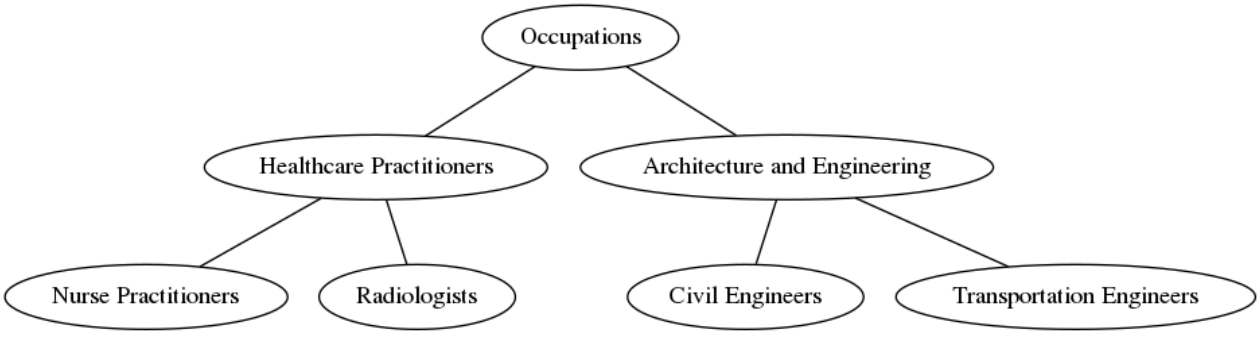
Referring to something as a skill, depending on the audience, may denote different subsets of competencies, or just as a shorthand for any competency. O*NET (https://www.onetonline.org/), for instance, uses 'skill' to refer to developed capacities (e.g. active listening). ESCO (https://ec.europa.eu/esco/portal) uses the terms 'skill' and 'competence' interchangeably. Other audiences may understand 'skill' to refer to some other subset of competencies.



The Skills-ML library is aimed at working with all types of competencies, as all of these are useful pieces of information about the labor market. *Competency* will be used throughout the rest of this paper for simplicity and clarity.

Occupation

An occupation is a normalized job title (e.g. Nurse Practitioner) defined in a competency framework. Normally, each occupation is assigned to an ID. The occupation ID is sometimes, but not necessarily structured in a hierarchical way. For instance, the coding structure of O*NET occupation is the Standard Occupational Classification (SOC) system which is a federal statistical standard used by federal statistical agencies to classify workers and jobs into occupational categories. In O*NET, occupations are categorized into different major groups as illustrated in the graph below. For example, Nurse Practitioner and Civil Engineer are occupations, but belong to Healthcare Practitioners and Architecture and Engineering, respectively.



Skill and Competency Contexts

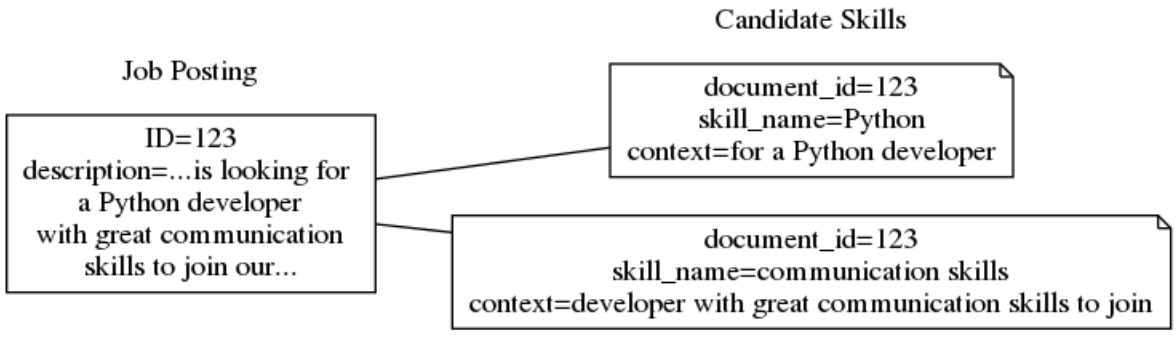
A skill/competency context describes the occurrence of a competency in some unstructured piece of text, such as a job posting or course description. Finding these occurrences is useful

for many tasks, such as analyzing the demand for competencies in job postings or the available opportunities to learn competencies in college courses.

Finding these occurrences is not easy, as there are numerous pieces of metadata regarding these occurrences that are important to track for a proper analysis.

Most important is the context (e.g. the sentence or paragraph in which the occurrence resides). It enables a human or an algorithm to determine if an occurrence is a true reference to a competency. For example, job postings routinely include links to social media pages such as Facebook, Twitter, or LinkedIn associated with the hiring company. If the context of these social media links were not taken into consideration, then it would appear that all job postings would require Facebook, Twitter, and LinkedIn as competencies. However, managing company content on social media is itself a competency for some occupations such as a Social Media Manager. The context can help disambiguate these from true references to competencies.

There are other useful pieces of metadata to attach to these occurrences, such as a reference to the matched competency in the reference ontology (if any), a reference to the method of extraction, and the original document. This metadata for each occurrence is contained in an object that Skills-ML calls a *CandidateSkill*.

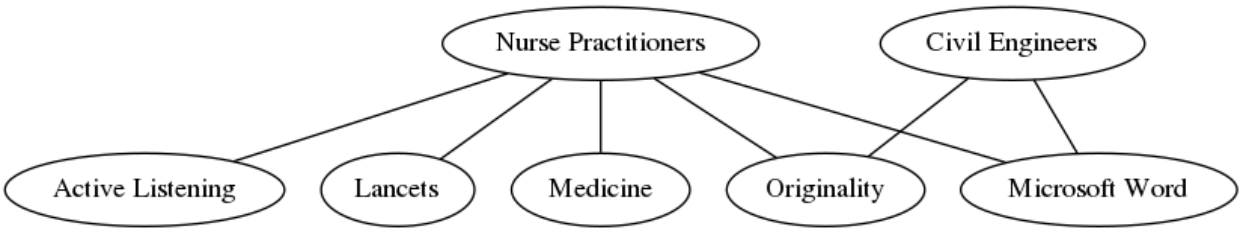


Ontologies

Essentially, an ontology is a knowledge graph to limit complexity and organize information into data and knowledge, encompassing a representation, formal naming, and definition of the categories, properties, and relations for certain domain knowledge. It is a concept that can be implemented in different ways. An ontology in Skills-ML specifically represents a collection of competencies, occupations, and all relationships between competencies and occupations.

There are a couple of different competency framework resources that have their own defined skills, abilities, knowledge and tools for each occupation, but they are not necessarily ontological, nor do they have their own ontological structures. For example, O*NET itself is not ontological and ESCO has its own defined ontology. One task in Skills-ML is to map the non-ontological or pre-defined ontological structure to the Skills-ML's *CompetencyOntology* structure which is implemented by

JSON-LD, and based on Credential Engine's CTDL-ASN format for Competencies, which is more flexible and can be used commonly throughout the Skills-ML library.



Implementation

This section details the capabilities that Skills-ML gives to researchers. The methodology and options for use are discussed heavily. For code examples and user instructions, refer to the library documentation. Terms formatted with italics are used throughout this section to denote specific classes or functions within Skills-ML.

Open Standards for Input and Output

Open standards are used as input and output as often as possible to serve the greater goal of interoperability between Skills-ML and other work in the workforce data space.

Input

Skills-ML makes use of schema.org's JobPosting standard. As it has been in use for a long time, some open sources are already using this standard, which is easy to import. Other job posting data sources are converted into the schema.org Schema and all work on job postings is done using this standard schema.

Users of Skills-ML who convert any job posting source they have access to into this schema can use any related functionality within the library whether it is stored locally, in the cloud, or in memory.

Output

When possible, JSON is used as an output format. Ontologies can be stored and retrieved as JSON-LD with a particular goal of interoperability by using the emerging CTDL-ASN standard for competencies. Candidate skills and evaluation metrics are returned as very simple data structures than can be easily serialized, in addition to convenience utilities that store them as JSON.

Modeling outputs that would be too complex to store with a simple format such as JSON are stored in the native format used by the underlying libraries. Word embeddings are stored as Gensim models and classifiers are stored as scikit-learn pickles.

Tabular output is also utilized when beneficial for the audience. The convenience functions for running document collections through various Skills-ML algorithms and aggregating their output data into CSV format to be easily used by researchers in the social sciences.

Generating Ontologies

Basic Structure

The structure of ontologies borrows a lot from graph theory, and in this section we'll use terminology from graph theory such as 'nodes' and 'edges'. For instance, when considering competencies, each competency is a 'node' and each relationship between competencies is an 'edge'. Ontologies are represented as undirected graphs, and in the library are known as *CompetencyOntology* objects. Each node in a *CompetencyOntology* can be either a *Competency* or *Occupation* (and in the future could be expanded to include other types of nodes), and edges represent relationships between two competencies, two occupations, or an occupation and a competency.

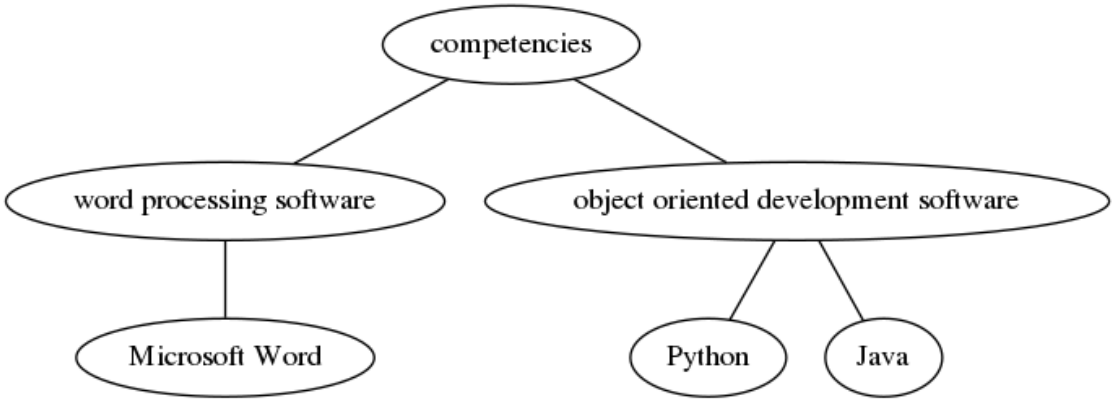
In keeping with the CTDL-ASN standard, the nodes themselves store the edges to other nodes of the same type. This is so the collection of nodes of a single type (say, all *Competencies*) is an independent framework usable in its own right, with all of the relations between different competencies intact.

The *CompetencyOntology* object itself contains the edges between nodes of different types. The existence of an edge between a *Competency* and an *Occupation* node signals that the *Competency* is relevant to the *Occupation*.

Hierarchies

Nodes of the same type have some special semantics for defining parent/child relationships as this is a very common relationship necessary to express existing competency and occupation frameworks. A node defined as a parent generally is a broader version of all of its children, having many shared attributes. For instance, ESCO defines an 'advanced nurse practitioner' and 'specialist nurse' as both being children of 'nursing professionals'. These occupations understandably share many competencies, and it is easy to imagine experience in any type of nursing professional occupation as being broadly applicable to other nursing professional occupations.

The same hierarchy is expressible in competencies; ‘proficiency with object-oriented programming software’ can be a competency that has many children, such as ‘proficiency with programming in Python’ and ‘proficiency with programming in Java’. Although they are distinct competencies, experience with one object-oriented programming language typically lessens the burden of learning another object-oriented programming language.



This parent/child hierarchy is necessary, but not itself sufficient for defining rich ontologies capable of expressing the relationships competencies can share. It can be useful to define relationships between competencies with different parents, or more rich relationships between competencies of the same parent. For this reason, it is possible to add other edge attributes. CTDL-ASN defines a number of ways to align concepts with each other, and each Competency node can be used to store these alignments as well as any other desired alignments.

Generation

CompetencyOntology objects can be generated programmatically by users of the Skills-ML library, either by iteratively calling methods to add nodes and edges, or by importing a JSON-LD file that defines the ontology. The semantics assumed by the *CompetencyOntology* JSON-LD importer are aimed at compatibility with other competency frameworks that follow the CTDL-ASN format. Although CTDL-ASN is only scoped for competencies and thus is not 1-1 compatible with a *CompetencyOntology* object, the competency portion of the *CompetencyOntology* is encapsulated by an object known as a *CompetencyFramework*, which can be used whenever competencies are the only node type needed.

A *CompetencyOntology* can also be produced algorithmically, from a list of *CandidateSkills*. If a competency extraction method, or ensemble of methods, is trusted enough to produce an ontology, this can be done automatically. Each competency name represented by the list of *CandidateSkills* is assumed to be a competency, and if there are occupations present in the source documents, those occupations will be added to the *CompetencyOntology* linked to the competencies they co-occur with.

Intrinsic Occupation Clustering

Clustering

Unlike competency, an occupation is defined as a leaf in a disjoint tree, which is less complex, but it still has the hierarchy structure. All the non-leaf nodes are some kind of classification determined by domain experts in different levels. For example, O*NET has 23 trees where each tree is a major group. It starts from the top node as a major group and splits into minor groups. Each minor group is broken into broad occupations, which is broken into detailed occupations. Finally, the leaves are the final SOC codes representing occupations with no children. A cluster here can be defined as leaves collapsed in different levels of a tree. In other words, those occupations who share the same parent or ancestor are in the same cluster. Therefore, an occupation tree is a cluster and there are clusters within clusters. Note that the tree structure of ESCO is very similar to O*NET, but it is deeper and denser.

Skills-ML has predefined two intrinsic clusterings for the O*NET ontology model.

1. MajorGroup - Occupation: Cluster concepts are major groups and cluster members are the occupation leaves of each major group tree.
2. MajorGroup - Competency: With the same major group clustering, one can collect the competencies associated to member occupations of a cluster. Therefore, cluster concepts are major groups and cluster members are the competencies collected from all the occupations leaves of each major group.

Hierarchical Distance

Since each digit in the SOC code in O*NET or the occupation code in ESCO means some kind of split in an occupation tree, a distance metric can be defined between two occupations as the total number of steps to the common ancestor. For example, the occupation tree in O*NET has

5 layers. Within the same tree, the maximum distance of two occupations is 4 steps. Since each occupation tree does not have a link with other occupation trees, two occupations from different trees will have a distance of 5. The caveat of this distance is its lack of taking inter-class information into account. In other words, if two occupations are in different occupation trees, the distance is the maximum distance no matter how the occupations may be related to each other. For example, the distance between Physicists (19-2012) and Computer and Information Research Scientists (15-1221) is the same as the distance between Physicists (19-2012) and Graphic Designers(27-1024), which does not really capture the true relationship between each occupation pairing. Therefore, a more well-defined distance metric is needed to better describe the relationship between the occupations.

Mixing Occupation and Competency Data for Clustering

One metric to determine the similarity of two occupations is to look at the common competencies shared by them. If two occupations shared a lot of competencies, even though the names of two occupations are very different or their locations in the occupation tree are very far from each other, they will be viewed as similar occupations. Therefore, the distance metric will be defined as the complementary of the similarity metric. The jaccard distance will be used as our distance metric, which is equal to 1 - jaccard similarity. In terms of the occupation and competency, the jaccard similarity of two occupations is defined as the number of common competencies divided by the number of total distinct competencies from two occupations. Since there exists a distance between any two occupations, a distance matrix for any ontology can be generated which can then be used to find the occupation clusters. This distance matrix can be used with any number of different clustering algorithms.

Labeling and Dictionary-Based Matching

Overview

There are multiple existing ontologies available for use, and there are many types of analyses involving matching terms from these ontologies within large amounts of unstructured text, such as job postings. A community college may want to see which set of competencies is the most in demand within their metropolitan area in order to create new courses. An economist studying automation may want to see which competencies have been declining in importance over time. A researcher looking for emerging competencies may want to use an existing ontology as a base to add new terms. Skills-ML aims to make it as easy as possible for researchers to find occurrences of these terms within unstructured text in an efficient manner.

Matching

Skills-ML uses different methods for finding occurrences of known terms in unstructured text.

- Exact matching finds an exact occurrence of a term in a *CompetencyFramework*
- Fuzzy matching finds occurrences of *CompetencyFramework* terms within a reasonable (and configurable) edit distance[1], in order to catch misspellings or similar forms of these terms within the text. To make this procedure manageable for a large set of documents,

1https://en.wikipedia.org/wiki/Edit_distance

the symspell algorithm is used with a sliding window of n-grams, which comes up with different results than exact matching if searched terms do not match up with word boundaries in the text (e.g. searching for ‘sledding’ in a document that contains the string ‘skating/sledding’).

- Occupation-scoped exact matching applies prior knowledge of competency/occupation matches encoded in *CompetencyOntology* to scope its search for terms that are known to be relevant for the occupation associated with the text. This provides a high-precision approach to implement exact matching, at the expense of a good amount of recall.

Evaluation

It is important to have a way to compare the results from different matching strategies. Expert-labeled ground truth data, or a ‘gold standard’ dataset, are always the ideal goal, to verify whether some *CandidateSkill* produced by an algorithm truly represents a competency in that context. Although precision in this sense is important, it is not the only useful metric, and it can be useful to generate metrics on bodies of text that are too large to reasonably produce gold standard labels. For this reason, Skills-ML implements a variety of metrics, some that only require a collection of *CandidateSkills* in order to compute.

- *TotalOccurrences* simply counts how many candidate skills were generated for the sample. This is a basic ‘sanity check’ metric and is most useful in conjunction with other metrics.
- *TotalVocabularySize* counts how many distinct competencies were found. This gives the researcher an idea of the breadth of the extraction results. If the number is too close to *TotalOccurrences*, the extractor may be producing matches that are too detailed and do not repeat. If the number is too small, the extractor may only be successful on a small, biased sample of desired terms.
- *PercentageNoSkillDocuments* computes the percentage of documents that failed to produce a candidate skill. If the researcher’s goal is to apply an extraction method that is useful for most documents, a high value for this metric would be concerning. When comparing this value across many different skill extractors, a high minimum value may indicate a problem with the document sample more than a problem with the skill extractors.
- *MedianSkillsPerDocument* computes the median number of candidate skills per document, which aims to measure skill density.
- *SkillsPerDocumentHistogram* is a more detailed version of PercentageNoSkillDocuments and MedianSkillsPerDocument. Using a fixed number of bins, this function computes the number of skills per document for each percentile bin and shows an even more detailed picture of skill density.
- *OntologyCompetencyRecall* takes an ontology and the candidate skills to compute the percentage of competencies in the ontology that were present at least once in the sample. If a sample appears to be skill-dense, a low value here may indicate some problems with the ontology, or a competency bias on the part of the skill extraction method.

- *OntologyOccupationRecall* takes an ontology in addition to the candidate skills and computes the percentage of occupations in the ontology that have a candidate skill present. This is only useful for documents that include an occupation. It is important to note that this is different than just looking at the occupations in the sample. This function takes into consideration the occupations with skills in the sample. This can provide a view into whether or not the skill extraction method is biased against different occupations.
- *GoldStandardPrecision* takes two sets of candidate skills from the same sample: one to interpret as ground truth, the other to evaluate against the ground truth. The function calculates the percentage of candidate skills in the evaluation list that are also present in the gold standard list.
- *GoldStandardRecall* takes two sets of candidate skills from the same sample: one to interpret as ground truth, the other to evaluate against the ground truth. The function calculates the percentage of candidate skills in the gold standard list also present in the evaluation list.

Labeling

CandidateSkills have thus far been discussed in the context of algorithms, but can also be produced by humans. Skills-ML does not include any interface for human labeling, but does include import and export utilities for interfacing with BRAT labeling software.

Representation Learning

Overview

Labor market data tends to be large in scale, but represented as raw text. Consequently, an important early step for most tasks is to transform texts into a mathematical form that can be used in the downstream tasks. Representation learning is good for this purpose. Representation learning is a general term for techniques that allow a system to automatically discover the representations needed for feature detection or classification from raw data. In Natural Language Processing (NLP), it is also recognized as vector space model (VSM) or word embedding. Word embeddings or vector space models are a more informative way of representing words, sentences or documents in a vector form, compared to one-hot encodings. VSMs have a long, rich history in NLP, but all methods depend in some way or another on the Distributional Hypothesis, which states that words that appear in the same contexts share semantic meaning.

Why Embedding?

Embeddings that are pre-trained on a large corpus can be plugged into a variety of downstream task models (e.g. sentiment analysis, classification) to automatically improve their performance by incorporating some general word/sentence representations learned on the larger dataset. In the context of skills and jobs, an embedding model trained on large amount of job posting data is able to map a skill or a job title into a high dimensional space as well as preserving the contextual and semantic relationship. Ideally, a good embedding model will cluster similar skills and jobs.

Embedding Model Pool

Many word embedding techniques have been developed since *word2vec*, the most impactful embedding algorithm, was published in 2013. Currently, Skills-ML includes *word2vec*, *doc2vec* and *fastext* and may include more in the future. For more details of how each model works, please see each paper's link.

- *Word2VecModel* is able to look up a word vector and infer a sentence/paragraph vector by averaging each word in a sentence/paragraph. It supports online learning. For out-of-vocabulary word handling of sentence/paragraph inference, a random vector will be assigned with the same dimension.
- *Doc2VecModel* is able to look up a word vector and infer a sentence/paragraph vector by gradient descending on the fly, so it is non-deterministic. It does not support online learning.
- *FastTextModel* is able to look up a word vector and infer a sentence/paragraph vector by averaging each word in a sentence/paragraph. It supports online learning. For out-of-vocabulary word handling of sentence/paragraph inference, it sums all vectors of the unseen word's char-ngrams. If none of the char-ngrams of the unseen word is present, a random vector will be assigned with the same dimension.

Embedding Training

Skills-ML includes online learning for training an embedding model if the selected model allows. The *EmbeddingTrainer* is able to specify the corpus, the model and the batch size to train an embedding model. A lookup dictionary of training documents could be saved by toggling the input argument lookup of the method train().

Embedding Model Evaluation

Although there is an emerging trend toward generating embeddings for structured and unstructured data, there is not yet a systematic suite for measuring the quality of embeddings. We generally follow one of the few works in embedding evaluation [Concept2vec: Metrics for Evaluating Quality of Embeddings for Ontological Concepts] to create metrics for evaluating embedding against the gold standard ontology dataset. The gold standard ontology is curated by domain experts like O*NET, so a good embedding should replicate the structure of the entities in the gold standard taxonomy. In other words, it is useful to see how an embedding reflects the clustering structure. Currently, Skills-ML only implements the most straightforward clustering approach: occupation major groups.

- *CategorizationMetric*: The cosine similarity between the embedding of the concept and the mean vector of embeddings of all the entities within that concept cluster. This metric aligns a clustering of entities into different categories, reflecting how well the embedding of a concept cluster performs as the background concept of the entities typed by it.
- *IntraClusterCohesion*: The sum of squared error of the embedding of the centroid of the concept cluster and the embedding of each entity within that cluster. It measures how near the data points in a cluster are to the cluster centroid.

- *MajorGroupRecall*: For a major group, calculate the cosine similarity against all the occupations and find the top n closest occupations. The recall is defined as the number of true positives from top n closest occupations divided by the total number of occupation within the major group.
- *MajorGroupPrecision*: Similarly to *MajorGroupRecall* which is called *Coherence Score* in the paper, start by finding the top n closest occupations. The precision is defined as the number of true positives from top n closest occupations divided by n

Each metric is calculated for each cluster to understand which cluster is reflected well by the embedding. The mean, variance, max, and min of each metric across all clusters give a global view of the embedding performance.

Classification

Overview

A common issue with job posting data is incomplete, incorrect, and inconsistent occupation classification. The majority of job postings in the US use the O*NET SOC classification system, but many are either missing or poorly classified. This can be improved by using machine learning.

SOC Codes

Most of the job posting data collected are aligned with the O*NET SOC system. The occupations in the SOC are classified at four levels of aggregation: **major group**, **minor group**, **broad occupation**, and **detailed occupation**. Each lower level of detail identifies a more specific group of occupations. Each item in the SOC is designated by a six-digit code. The first two digits represent the major group, the third digit represents the minor group, the fourth and fifth digits represent the broad occupation, and the sixth digit represents the detailed occupation.

- Major group codes end with 0000 (e.g., 29-0000 Healthcare Practitioners and Technical Occupations —the exceptions are minor groups 15-1200 Computer Occupations, 31-1100 Home Health and Personal Care Aides; and Nursing Assistants, Orderlies, and Psychiatric Aides, and 51-5100 Printing Workers, which end with 00).
- Minor groups generally end with 000 (e.g., 29-1000 Health Diagnosing or Treating Practitioners).
- Broad occupations end with 0 (e.g., 29-1020 Dentists).
- Detailed occupations end with a number other than 0 (e.g., 29-1022 Oral and Maxillofacial Surgeons).

Classification

The classification task consists of inferring a SOC code from a job posting and is accomplished through several stages: preprocessing, filtering, training, and testing. Skills-ML allows the user to train multiple models and tune hyperparameters in an experiment by configuring them in a grid. The following Python classes help the user accomplish this task.

- *IterablePipeline* helps compose a sequence of operations or processing and can be passed around between stages.
- *SocEncoder* converts a 6-digit SOC code to a class number.
- *SOCMajorGroup* specifies the target variable to be the major group which is inherited from the *TargetVariable* class that specifies what target variable we want to predict/optimize for the model. It also filters the job posting based on the criteria it takes in.
- *FullSOC* is similar to *SOCMajorGroup,* but specifies the target variable to be the full SOC code.
- *DesignMatrix* takes in a data source, a pipeline for training data, a pipeline for target variable, and a specified target variable as a container to be fed into the trainer.
- *OccupationClassifierTrainer* trains classifiers with cross validation and picks the best classifier with a grid search based on the metric. It takes in a dictionary for the grid search, a *DesignMatrix* specifies the x and y, and the number of folds for cross-validation.
- *CombinedClassifier* is a classifier that takes in an embedding model and a classifier as a combined classifier.
- *KNNDoc2VecClassifier* is a KNN classifier based on Doc2Vec embedding model.

Evaluation
Once we have trained a group of classifiers, we want to evaluate the performance of each classifier on the test dataset. Accuracy, recall, precision, and f1 are the metrics taken into consideration. Since it is a multi-class classification problem, an overall performance is evaluated by looking at the micro-average and macro-average for the metrics. A macro-average will compute the metric independently for each class and then take the average, whereas a micro-average will aggregate the contributions of all classes and then compute the average. In other words, a macro-average is treating all classes equally. If the test dataset appears to be class imbalanced, then the micro-average of metrics will be used.

Experiments and Results

This section describes the results from applying various Skills-ML algorithms to public and private data sources.

Job postings used for Skills-ML span the time period of 2006-2018 with most postings occuring in the last few years in the United States. These postings were provided by the National Association of State Workforce Boards. Different sample sizes were used in different contexts, the details of which are covered in specific sections. Although the source data is not publicly available, there are no comparable public datasets to use at the time of publication. All public alternative datasets are either too small or too geographically limited in scope to effectively serve as an example. Users wishing to run these algorithms can do so with any public or private

job posting datasets they have available.

The source data to create the O*NET and ESCO ontologies were downloaded from their respective public web sites, and the code to download and create them is available through the Skills-ML module.

Ontology Generation

Several ontologies were created using a mixture of pre-existing survey-produced standards and derived lists from a sample of job postings across the United States.

The job posting sample consisted of 100,000 postings from 2006-2018, heavily biased towards 2015-2017. The job postings were tagged with 863 unique O*NET SOC codes out of a possible 1,133.

ONET - KSA consists of O*NET Knowledge, Skills, and Abilities. This list represents a more general list of high-quality competencies that are heavily transferable between occupations.

‘Nurse Practitioners’ O*NET Knowledge, Skills, and Abilities sample competencies: ‘Active Listening’, ‘Medicine and Dentistry’, ‘Oral Comprehension’

ONET - T2 consists of O*NET Tools and Technology. The large Tools and Technology lists make this into a somewhat sparse ontology, as there are many such Tools and Technology competencies that are only associated with one occupation and/or are badly outdated.

‘Nurse Practitioners’ O*NET Tools and Technology sample competencies: ‘Office suite software’, ‘Eye Charts or Vision Cards’

ONET - DWA consists of O*NET Detailed Work Activities. These detailed work activities differ from the other O*NET definitions in that they are slightly more specific and phrased as verb phrases rather than noun phrases.

‘Nurse Practitioners’ O*NET Detailed Work Activities sample competencies: ‘Treat Medical Emergencies’, ‘Order medical diagnostic or clinical tests’

ONET - ALL consists of O*NET Knowledge, Skills, Abilities, Tools, Technology, and Detailed Work Activities. This flat list of differently-phrased and collected O*NET competencies is the only version of O*NET that matches ESCO in scope, and as such is the best candidate for evaluation directly against ESCO.

‘Nurse Practitioners’ O*NET All sample competencies: ‘Office suite software’, ‘Active Listening’, ‘Medicine and Dentistry’, ‘Oral Comprehension’, ‘Eye Charts or Vision Cards’, ‘Treat Medical Emergencies’, ‘Order medical diagnostic or clinical tests’

ESCO - ALL consists of the entirety of the ESCO (European Skills, Competences, Occupations) ontology. It includes the scope of ONET - ALL, combining academic knowledge, soft skills, tools, and tasks, but the list is more highly curated than O*NET's version so there are far fewer singular and outdated competencies. In addition, the phrasing of these competencies is often done using verb phrases instead of noun phrases. This may provide some differences in matching results depending on how authors phrase competency requirements in job postings.

'Advanced nurse practitioner' ESCO All sample competencies: 'advise on healthy lifestyles', 'have computer literacy', 'listen actively'

Derived - skill phrases consists of competencies generated from scanning the sample of job postings and extracting all noun phrases ending with 'skill', 'skills', 'ability', or 'abilities'. These competencies are associated in the ontology with the given O*NET SOC code for the job posting if it exists. The scope of this is highly dependent on the size of the sample. The sample itself contained 863 unique occupations, but only 560 occupations had any skill phrases successfully extracted so it represents a very incomplete picture of the occupation space. However, because of the fairly tight filter around what gets returned as a potential competency, the results appear to be highly relevant.

'Registered Nurse' Derived Skill Phrase sample competencies: 'analytical ability', 'dialysis skills', 'excellent communication skills', 'basic arrhythmia recognition skills', 'communication skills'

Derived - skill section consists of competencies generated from the 'skills' section of job postings. The procedure for computing this involves making inferences about the structure of a job posting and dividing it into sections based on lines that look like headers. Sections are filtered down to those tagged by a few common headers that imply competencies (such as 'skills', 'competencies', 'abilities'), and each sentence contained in the section is returned as a potential competency. The goal is to remove the vast amount of information about the hiring organization and other irrelevant information that often clutters job postings and focus on the sections discussing requirements for the job.

'Registered Nurse' Skill Section sample competencies: 'organizational competencies: recognizes the impact of systems on health care delivery.', 'ability to function effectively in a fluid, dynamic, and rapidly changing environment.', 'on off-shifts intervenes immediately with any employee sustaining an exposure to a bloodborne pathogen.'

Ontology	# of Competencies	# of Occupations	# of Competency-Occupation Relationships	Median Occupations Per Competency	Median Competencies Per Occupation
ONET - ALL	34,100	1,133	124,584	1	107
ONET - DWA	2,070	974	17,279	6	17

ONET - T2	31,909	974	68,689	1	55
ONET - KSA	120	966	38,616	231	41
ESCO - ALL	12,215	2,127	66,762	2	31
Derived - Skill Phrases	4,498	560	15,773	1	8
Derived - Skill Section	143,318	548	161,100	1	49

Matching

Various ontologies and matching algorithms are performed against a set of 24,000 job postings. The results are grouped by a matching algorithm. The dataset does not have labels so a reliable precision calculation is not available.

Below are some important points illustrated by the results. The full matching results tables are contained in Appendix B.

Fuzzy matching does not offer vast improvements over exact matching for any ontologies. The closest is the experimental *Derived - Skill Phrases* ontology, whose recall increased from 0.37 to 0.43. This change might reflect the messy nature of the ontology, whose results are gleaned from the job posting data and not heavily cleaned. As noted in the Implementation section, the fuzzy matcher uses word tokenization, which means that the exact matcher recalls some examples that cross tokenization boundaries that the fuzzy matcher will not (e.g. ‘business/systems analysis’ if the term being searched for is ‘systems analysis’), which surprisingly makes the results for exact matching slightly higher than fuzzy matching in some cases.

The impact of occupation-scoping the matcher is most heavily observed in the *ONET - T2* and *Derived - Skill Section* competency frameworks. In the case of ONET - T2, this illustrates a problem seen often by the team, in that tools or technology with short names (e.g. the old programming language ‘forth’) that frequently show up in plain English, may not necessarily indicate a skill listed in the job posting. Social media websites such as Facebook and Twitter show up as legitimate competencies for a handful of occupations, but show up in a large proportion of job postings as company metadata. Occupation-scoping the matcher is aimed at fixing this problem, and it decimates the recall of *ONET - T2* and thus *ONET - All. ONET - KSA*’s recall is reduced, but much less drastically so. The *Derived - Skill Section* framework, as one may guess with 300,000 matches on a 24,000 document dataset, is likely inundated with low-quality matches representing generic sentences. *Derived - Skill Phrases* actually receives the smallest reduction in occurrences from this change. Occupation-scoping the matcher for *ESCO - ALL* is not possible with this dataset as the job postings are not labeled with ESCO occupations.

The *ONET - DWA* competency framework is nearly useless when matched using any of the available algorithms. Although at first glance its competencies are phrased similarly to those in ESCO, the phrases used in ESCO are far more likely to show up in job postings. It's not immediately clear why this is the case, but ESCO phrases may be slightly more concise or simply phrased in a manner more likely to mirror the writing style of job postings.

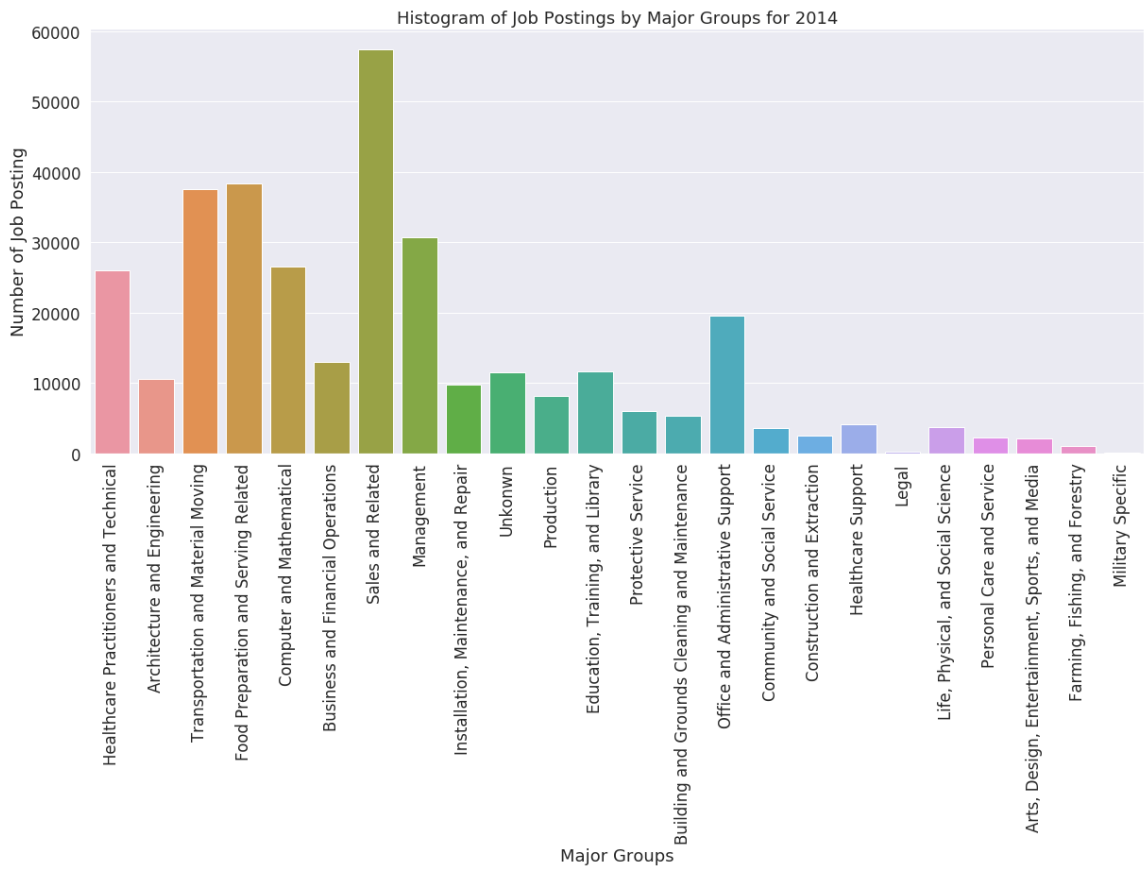
The *ONET - KSA* competency framework is well-recalled in the dataset. This is not terribly surprising given the small size of the framework and relatively large size of the dataset.

The *ONET - All* and *ESCO - All* competency frameworks performed similarly, excepting occupation recall. The occupation recall for O*NET was almost twice that of ESCO. This may indicate some type of bias on ESCO's part, but could also reflect the fact that about twice as many occupations are available in ESCO as in O*NET, requiring a larger (or at least more diverse) sample size of job postings to reproduce.

The poor competency recall of *Derived - Skill Section* given the high number of occurrences is worthy of note, even if this behavior is not shocking given the size of the ontology or its method of production. There are a lot of competencies in this ontology that look highly relevant in isolation but are unique to their source job posting and don't transfer well to a new sample. This indicates that effort to clean and consolidate the results of the skill section extraction into a form more commonly transferable to other postings would be useful.

Word Embedding

To demonstrate the experiments of word embedding, we include 6 types of embedding models - Doc2Vec-Distributed Memory, Doc2Vec-Distributed Bag of Words, Word2Vec-SkipGram, Word2Vec-Continuous Bag of Words, FastText-SkipGram, and FastText-Continuous Bag of Words for training. The only hyperparameter tuning here is the window size, (range of 5, 7, or 13), which controls how wide the gap between two items in a sequence can be so that they are still considered in the same context. The rest of the hyperparameters are the same: vector size=300, number of epoch=10, number of negative sample=5, and initial learning rate=0.025. All of the embeddings were trained on 2014 job posting data from the National Labor Exchange, which includes 330,000 job postings with the major group distribution in the graph below. We then evaluate embedding models by looking at how well they reflect the structure of the major group occupation clusters. Each cluster is a major group and each entity is an occupation. We represent the major group name in the embedding space as a cluster concept and represent the occupation name and description in the embedding space as a cluster member.



The goal is to choose an embedding model by looking at all 4 metrics and use that embedding model in the occupation classification later. We found that the Doc2Vec models are generally worse than the rest in *CategorizationMetric*. FastText-CBOW and Word2Vec-CBOW are the best for *CategorizationMetric*. For *IntraClusterCohesion*, it is only reasonable to compare the same model with different window sizes. The smaller the value, the better the clustering capability. Only two types of models have a huge difference in the window size in the *IntraClusterCohesionby*. FastText-CBOW with a window size of 13 is better than a window size of 5 and 7. Word2Vec-CBOW model with a window size of 5 is better than a window size of 7 and 13. For *PrecisionTop30* and *RecallTop30*, all models are generally bad. The best model is the Word2Vec-CBOW with a window size of 5. By taking all of the metrics into account, the Word2Vec-CBOW model with a window size of 5 was the best embedding model.

Occupation Classification

The next task is occupation classification, or classifying a job posting with a predicted O*NET SOC Code by using the context of job postings. The data used in this task is a curated job posting dataset that has a similar amount of job postings from each major group. Since this is a multi-class supervised machine learning problem, the label is the O*NET SOC Code for each job posting and there are 690 different labels in this case. By choosing the best embedding model as a job posting vectorizer from the previous experiment, we trained 4 types of classifiers

on top of it, ExtraTrees, RandomForest (RF), Multi-Layer Perceptron (MLP), and Support Vector Machine (SVM). Note that the best embedding model doesn't mean it would be the best candidate vectorizer for classification. With grid search and 3-fold cross-validation, we obtained the best classifiers for each type.

The table below shows the comparison of different classifiers with tuned hyperparameters. Note that micro-precision is higher than macro-metrics in general, this is because of the high variance of performance of each class. Even though the curated job posting dataset is fairly balanced across all major groups, we found the full O*NET SOC Code is still imbalanced. The best classifier is MLP with the micro-averaging precision 0.6451, hidden layer size 500 and logistic function as activation function. In general, the performance suffers due to the large number of classes and imbalanced O*NET SOC code labels in the training data that results in poor performance in several classes. However, if we only look at the major group (first 2 digits) in the predicted O*NET SOC code, the accuracy is boosted for each type and for MLP is 0.7951.

	ExtraTrees	RF	MLP	SVM
micro-precision	0.4870	0.4970	0.6451	0.6329
macro-precision	0.3067	0.3007	0.3678	0.3497
macro-recall	0.1764	0.1772	0.3284	0.3200
macro-f1	0.1987	0.1972	0.3289	0.3192
micro-precision major group	0.6132	0.6316	0.7951	0.7775

Case Studies: Economic White Papers

There are two in-progress economic white papers based on the job posting data processed through Skills-ML.

'The Costs of Occupation Switching', by Nayoung Rim at the United States Naval Academy and Nuno Paixao at the Bank of Canada, uses job posting data processed through Skills-ML to analyze the possible reasons for occupation switching. Competencies extracted from the job postings in specific geographic areas are used as an indicator of demand for those competencies. Lack of demand for these competencies is used as one possible reason for switching occupations.

'The Effect of Additional Authority on Job Postings: The Case for Nurse Practitioners', by Sarah Bana at University of California - Santa Barbara, uses job posting data processed through Skills-ML to analyze how demand for Nurse Practitioners is related to state-level policy changes regarding the authority of Nurse Practitioners. Counts of Nurse Practitioner job postings in specific states over particular time periods are used as an indicator of demand for Nurse

Practitioners. Any correlations between large changes in demand for Nurse Practitioners around the times of policy changes are analyzed.

Architecture

Skills-ML design philosophy builds on dataflow programming or so-called data streaming to process very large datasets (larger than RAM; potentially infinite). Data points are processed one at a time, in constant RAM. This places some requirements on the algorithms and methods used like generator, online learning, reservoir sampling, and single pass methods to allow people to use it not only on the cloud, but also locally without huge infrastructure requirements.

Generator

One technique that is used throughout Skills-ML for data streaming is generator. A generator, in a nutshell, is a function or object which produces a stream of values over the course of its lifetime. The advantage of a generator is to avoid the memory problem, which means inability to store data in one variable. The lazy evaluation of generator gives a value only when it is asked, which means only that single value takes up memory and makes it a perfect candidate for reading and using “Big Data” files very memory-efficient. In Skills-ML, normally any data source or collection is a generator for propagating into different components. For example, *JobPostingCollectionFromS3* is a generator that streams one job posting at a time from the s3 bucket.

In addition to the data source and collection, most of the processing and transformation are meant to build upon generators in Skills-ML. When users want to apply some transformation on a data collection, they can chain many of these transformations together. For example, when you train a word2vec embedding model, it also expects to see the training data as a generator. Also, *JobSampler* is used to randomize a sample from a data stream using reservoir sampling algorithm.

Storage

The storage object in Skills-ML allows code to refer to a large object without storing it in system memory. Most of the classes have an argument for storage object, especially models and classifiers. It is common to combine two classifiers into one and store it as the final model, or to store the whole pipeline with models and classifiers in the experiment. A complicated embedding model trained on a very large corpus could take a couple of gigabytes to store so if we want to reuse the model and store it somewhere else, we do not want to actually store the model again. The storage object refers to the original model instead of saving a copy. Therefore, it is resource-efficient and compact. To benefit both cloud-based and local usage, storage class was created for both usages.

Aggregator and Computer

Working with large amounts of data often requires a fair amount of aggregation by the researcher to gain any sort of insights. Aggregation is hard to perform online, row-by-row without resorting to custom code or heavy map-reduce frameworks that require a tremendous

amount of setup, administration, and hardware. To allow researchers with a laptop and teams with the ability to set up computing clusters to perform aggregation in roughly the same way, Skills-ML provides the Computer and Aggregator classes, which can be used as a basic storage-backed map-reduce framework for collections of documents such as job postings. The framework allows the researcher or team to:

1. Partition the document collection according to deterministic document attribute, such as a posting date or a certain number of digits of the document's unique identifier
2. Compute some number of properties of each document (such as the skills produced by a specific skill extractor, or the occupation produced by a specific occupation classifier), serializing each document's properties to storage (e.g. local filesystem or Amazon S3) according to the partition
3. Assemble different tabular aggregations of these serialized computed properties without needing to recompute the properties

This process can be parallelized based on the user's preference. This enables a team with access to a computing cluster to heavily parallelize the computation of large datasets and serialize to cloud storage, or a researcher with access to only a laptop to run it on one or a handful of processes on their laptop serialized to a local disk using no extra libraries.

Related Toolkits and Platforms

Skills-ML relates to a number of similar or complementary efforts to automate various aspects of skill and competency ontology generation happening in other research groups as well as the private sector. Efforts such as Skillscape (Science, 2018) out of the MIT Media Lab are involved in analyzing competency relationships to explore skill transferability, and complement the work done in Skills-ML.

Future Work

Skills-ML can be improved in many ways. This section details next steps for any researcher looking to immediately build on the work in Skills-ML to make it more useful in a variety of areas.

Ontology Updating

Competency ontologies require an extensive amount of human labor to update. Being able to update an existing ontology using data extracted from a large, frequently updated text corpus such as job postings would help immensely. Recent research such as from Alsuhaibani, et al[2] explores methods for using both an ontology and a text corpus to dynamically expand the ontology. This approach could be utilized to mutate an ontology using new data over time.

Matching

2https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0193094

Competency statements in documents and survey-based ontologies can vary greatly in terms of how they are phrased. Noun and verb phrases of varying specificity are encountered. As the length of competency statements increases, the user might expect the probability of that full phrase being present in the document text to decrease. Additionally, the fuzzy matching present in Skills-ML was not built to handle this. It was built to handle misspellings and other cases where an occurrence may only vary by a few characters. This is insufficient and a more advanced approach to matching based on word/phrase embedding similarity may be utilized to provide a matching algorithm more resilient to phrasing, or even tokenization (e.g. the 'business/ systems analysis' problem discussed earlier in the paper).

Section Extraction

The procedure of filtering out all sections of a document that don't match a whitelisted header (e.g. 'skills', 'competencies') is explored in the paper, but only for the purpose of raw ontology creation, a task which it performs poorly. A more suitable use may be for training higher-quality word embeddings that ignore company information, or in conjunction with NLP techniques that produce less unique matches.

Gold Standard Creation

A large-scale effort to create a gold standard competency labeling would enable better evaluation of matching algorithms. In addition, the labels can be used to train a model such as that described by Huang, et al[3]. The label import functionality described in Skills-ML can be used to utilize such labels in the library, but the human effort to label a large enough sample of documents with competencies would be a huge undertaking.

Clustering

In the current version of Skills-ML, we provide some generic and explicit clusterings in the ontology, such as major groups. However, there are much more complicated ways to uncover underlying structures in the ontology, which means there are many possible ways to segment the occupations and competencies. By defining different distance metrics between nodes in ontology, we might be able to cluster the nodes differently. A huge benefit for having different clusterings is to provide a complete test to evaluate embedding models, in order to pick the embedding model that can reveal the structures of ontology from different angles.

Conclusion

Filling this gap in open, translatable competency data has been the focus of multi-year efforts by Credential Engine, University Professional and Continuing Education Association (UPCEA), the Academic Genome Project, IMS Global, the Open Skills Project, and others. Education and training providers now have many of the foundational components and standards they need to begin creating and publishing educational competencies within a curriculum in a standards-based machine-actionable way. Yet, individual faculty and course creators lack a clear and

3https://arxiv.org/abs/1508.01991

practical solution for implementation. Recent advancements in AI, natural language processing, and machine translation provide a new way forward that can build on past and current competency framework efforts to dramatically accelerate their adoption and use in education. New AI-assisted processes embedded in instructional design workflows can help automate competency identification and normalization, creating rich, structured competency data that aligns with existing standards -- ultimately with little manual intervention. Skills-ML provides a framework using algorithms and standards to help fill the gap of open competency data.

Appendix A: Glossary

Competency - Any expertise or talent that is useful for a job.

Skill - Any expertise or talent that is useful for a job.

Occupation - A normalized job title.

CandidateSkill - An instance of a Skill/Competency found in some document (e.g. a job posting), with surrounding context

CompetencyFramework - A collection of Competencies and their relationships with each other

CompetencyOntology - A collection of Competencies and Occupations and their relationships with each other

O*NET - Short for Occupational Information Network, O*NET is a database of occupations and metadata (including skills, but also tasks and many others) about those occupations maintained by the US Department of Labor. The data is produced by surveys and partially updated every quarter.

SOC Code - Short for Standard Occupational Classification Code. An eight-digit code for an occupation, maintained by O*NET.

Major group - An occupation major group in the O*NET SOC system, represented by the first two digits of the occupation’s SOC code.

ESCO - Short for European Skills, Competences and Occupations, ESCO is a database and classification system covering skills, competencies, qualifications, and occupations, maintained by the European Commission.

Appendix B: Matching Results Tables

Exact Matching

Ontology	Total Occurrences	Total Vocab. Size	Percentage No Skill Documents	Median Skills per Document	Ontology Competency Recall	Ontology Occupation Recall
ONET - All	101,185	2,105	18.4	2	.06	.59
ONET - T2	63,522	2,010	28.9	1	.06	.67
ONET - DWA	8	8	99.9	0	.003	.008
ONET - KSA	37,793	87	44.4	1	.73	.62
ESCO - All	85,217	1,230	18.6	2	.10	.31
Derived - Skill Phrases	34,362	1,703	43.1	1	.37	1.0
Derived - Skill Section	302,686	9,230	3.5	10	.06	1.0

Fuzzy Matching

Ontology	Total Occurrences	Total Vocab. Size	Percentage No Skill Documents	Median Skills per Document	Ontology Competenc y Recall	Ontology Occupation Recall
ONET - All	90,419	2,711	21.6	2	.06	.59
ONET - T2	54,758	2,520	34.2	1	.06	.65
ONET - DWA	180	51	99.2	0	.02	.07
ONET - KSA	35,481	140	45.8	1	.75	.62
ESCO - All	85,112	2,148	19.4	2	.12	.31

Derived - Skill Phrases	108,087	3,477	42.9	1	.43	1.0
Derived - Skill Section	421,199	10,660	4.05	11	.03	1.0

Occupation-Scoped Exact Matching

Ontology	Total Occurrences	Total Vocab. Size	Percentage No Skill Documents	Median Skills per Document	Ontology Competency Recall	Ontology Occupation Recall
ONET - All	24,464	465	63.0	0	.01	.37
ONET - T2	5,506	393	85.7	0	.01	.28
ONET - DWA	2	2	99.9	0	.001	.002
ONET - KSA	18,958	70	69.6	0	.59	.41
Derived - Skill Phrases	20,539	690	57.8	0	.15	.63
Derived - Skill Section	32,070	4,510	57.9	0	.03	.48

Appendix C: Embedding Evaluation Tables

Categorization Metric

Model	Window	Mean	Std. Dev.	Max	Max Cluster	Min	Min Cluster
Doc2vec - DM	5	0.3936	0.0984	0.6440	Military Specific Occupations	0.2161	Food Preparation and Serving Related Occupations
Doc2vec - DM	7	0.3020	0.1006	0.6412	Military Specific Occupations	0.1547	Construction and Extraction Occupations
Doc2vec - DM	13	0.1537	0.0829	0.4416	Military Specific Occupations	0.0537	Installation, Maintenance, and Repair Occupations
Doc2vec - DBOW	5	0.3581	0.0556	0.5050	Military Specific Occupations	0.2561	Food Preparation and Serving Related Occupations
Doc2vec - DBOW	7	0.3534	0.0544	0.5075	Military Specific Occupations	0.2630	Installation, Maintenance, and Repair Occupations
Doc2vec - DBOW	13	0.3562	0.0534	0.4835	Production Occupations	0.2669	Food Preparation and Serving Related Occupations
Word2vec - SG	5	0.3759	0.0938	0.5889	Production Occupations	0.255253	Building and Grounds Cleaning and Maintenance
Word2vec - SG	7	0.3682	0.0949	0.5746	Production Occupations	0.2361	Food Preparation and Serving Related Occupations
Word2vec - SG	13	0.3645	0.0954	0.5607	Production Occupations	0.2392	Building and Grounds

							Cleaning and Maintenance
Word2vec - CBOW	5	0.6279	0.1487	0.8728	Healthcare Support Occupations	0.3152	Installation, Maintenance, and Repair Occupations
Word2vec - CBOW	7	0.6197	0.1547	0.8561	Management Occupations	0.2983	Installation, Maintenance, and Repair Occupations
Word2vec -CBOW	13	0.6308	0.1592	0.9027	Healthcare Support Occupations	0.3167	Installation, Maintenance, and Repair Occupations
Fasttext - SG	5	0.3308	0.0866	0.5136	Production Occupations	0.2008	Food Preparation and Serving Related Occupations
Fasttext - SG	7	0.3180	0.0796	0.4874	Production Occupations	0.3180	Food Preparation and Serving Related Occupations
Fasttext - SG	13	0.2887	0.0759	0.4613	Management Occupations	0.1833	Building and Grounds Cleaning and Maintenance
Fasttext - CBOW	5	0.6535	0.1464	0.8870	Protective Service Occupations	0.3584	Installation, Maintenance, and Repair Occupations
Fasttext - CBOW	7	0.6761	0.1388	0.8936	Protective Service Occupations	0.3608	Installation, Maintenance, and Repair Occupations
Fasttext - CBOW	13	0.6968	0.1454	0.9027	Healthcare Support Occupations	0.3167	Installation, Maintenance, and Repair

							Occupations

Intra-Cluster Cohesion

Model	Window	Mean	Std. Dev.	Max	Max Cluster	Min	Min Cluster
Doc2vec - DM	5	274.37	160.74	562.47	Office and Administrative Support Occupations	52.60	Building and Grounds Cleaning and Maintenance
Doc2vec - DM	7	257.26	151.96	533.87	Office and Administrative Support Occupations	0.1547	Legal Occupations
Doc2vec - DM	13	246.73	147.22	528.11	Production Occupations	46.69	Legal Occupations
Doc2vec - DBOW	5	154.29	90.64	316.80	Production Occupations	25.12	Legal Occupations
Doc2vec - DBOW	7	154.44	90.72	317.5	Production Occupations	25.14	Legal Occupations
Doc2vec - DBOW	13	154.96	90.97	318.75	Production Occupations	25.03	Legal Occupations
Word2vec - SG	5	83.84	53.48	231.6	Production Occupations	13.10	Legal Occupations
Word2vec - SG	7	79.83	51.53	224.00	Production Occupations	12.36	Legal Occupations
Word2vec - SG	13	74.08	47.89	208.32	Production Occupations	11.57	Legal Occupations
Word2vec - CBOW	5	7100.51	4164.44	16298.7	Production Occupations	1223.83	Legal Occupations
Word2vec - CBOW	7	16419.1	9565.05	37113.3	Production Occupations	2910.32	Legal Occupations
Word2vec -CBOW	13	43978.5	25315.9	98805.3	Production Occupations	7716.23	Legal Occupations

Fasttext - SG	5	78.68	51.22	225.40	Production Occupations	11.91	Legal Occupations
Fasttext - SG	7	76.15	49.93	222.43	Production Occupations	11.84	Legal Occupations
Fasttext - SG	13	70.36	47.24	213.36	Production Occupations	11.35	Legal Occupations
Fasttext - CBOW	5	35989.1	21477.5	89178.1	Production Occupations	21477.5	Military Specific Occupations
Fasttext - CBOW	7	64812.9	38815	161705	Production Occupations	11736.3	Military Specific Occupations
Fasttext - CBOW	13	149895	89901.8	373883	Production Occupations	26673.5	Military Specific Occupations

Precision Top 30

Model	Window	Mean	Std. Dev.	Max	Max Cluster	Min	Min Cluster
aDoc2vec - DM	5	0.0608	0.0648	0.2000	Installation, Maintenance, and Repair Occupations	0	Community and Social Service Occupations
Doc2vec - DM	7	0.0478	0.0687	0.2666	Installation, Maintenance, and Repair Occupations	0	Community and Social Service Occupations
Doc2vec - DM	13	0.0550	0.0693	0.2000	Healthcare Practitioners and Technical Occupation	0	Community and Social Service Occupations
Doc2vec - DBOW	5	0.3406	0.2222	0.8333	Healthcare Practitioners and Technical Occupation	0.0333	Military Specific Occupations
Doc2vec - DBOW	7	0.3348	0.2306	0.8000	Healthcare Practitioners and Technical	0.0333	Military Specific Occupations

					Occupation		
Doc2vec - DBOW	13	0.3333	0.2387	0.8333	Healthcare Practitioners and Technical Occupation	0.0333	Military Specific Occupations
Word2vec - SG	5	0.2608	0.1369	0.6333	Architecture and Engineering Occupations	0.0667	Healthcare Support Occupations
Word2vec - SG	7	0.2478	0.1298	0.6	Architecture and Engineering Occupations	0.0667	Healthcare Support Occupations
Word2vec - SG	13	0.2348	0.1427	0.5333	Architecture and Engineering Occupations	0.0333	Management Occupations
Word2vec - CBOW	5	0.3231	0.1848	0.8333	Architecture and Engineering Occupations	0.0333	Healthcare Support Occupations
Word2vec - CBOW	7	0.2928	0.1801	0.7667	Architecture and Engineering Occupations	0.0333	Healthcare Support Occupations
Word2vec -CBOW	13	0.3173	0.1914	0.8	Architecture and Engineering Occupations	0.0333	Management Occupations
Fasttext - SG	5	0.2536	0.1681	0.6333	Architecture and Engineering Occupations	0.0333	Construction and Extraction Occupations
Fasttext - SG	7	0.2362	0.1623	0.6333	Architecture and Engineering Occupations	0.0333	Management Occupations
Fasttext - SG	13	0.2391	0.1598	0.6333	Architecture and Engineering Occupations	0.0333	Sales and Related Occupations

Fasttext - CBOW	5	0.2435	0.1662	0.7333	Installation, Maintenance, and Repair Occupations	0.0333	Construction and Extraction Occupations
Fasttext - CBOW	7	0.2435	0.1628	0.7	Installation, Maintenance, and Repair Occupations	0	Construction and Extraction Occupations
Fasttext - CBOW	13	0.2522	0.1632	0.7333	Installation, Maintenance, and Repair Occupations	0.0333	Healthcare Support Occupations

Recall Top 30

Model	Window	Mean	Std. Dev.	Max	Max Cluster	Min	Min Cluster
Doc2vec - DM	5	0.0379	0.050	0.1852	Sales and Related Occupations	0	Community and Social Service Occupations
Doc2vec - DM	7	0.0249	0.0305	0.1111	Sales and Related Occupations	0	Community and Social Service Occupations
Doc2vec - DM	13	0.0254	0.0234	0.0741	Sales and Related Occupations	0	Community and Social Service Occupations
Doc2vec - DBOW	5	0.2477	0.1949	0.8947	Food Preparation and Serving Related Occupations	0.0328	Production Occupations
Doc2vec - DBOW	7	0.2552	0.1917	0.8421	Food Preparation and Serving Related Occupations	0.0409	Production Occupations
Doc2vec - DBOW	13	0.2424	0.1678	0.7895	Food Preparation and Serving Related Occupations	0.0328	Production Occupations

Word2vec - SG	5	0.2386	0.1984	0.6333	Food Preparation and Serving Related Occupations	0.0409	Healthcare Support Occupations
Word2vec - SG	7	0.2227	0.1827	0.7368	Food Preparation and Serving Related Occupations	0.0298	Construction and Extraction Occupations
Word2vec - SG	13	0.2072	0.1659	0.6	Building and Grounds Cleaning and Maintenance	0.0156	Management Occupations
Word2vec - CBOW	5	0.2620	0.1777	0.7368	Food Preparation and Serving Related Occupations	0.0491	Production Occupations
Word2vec - CBOW	7	0.2482	0.1902	0.7368	Food Preparation and Serving Related Occupations	0.0435	Healthcare Practitioners and Technical Occupations
Word2vec -CBOW	13	0.2511	0.1799	0.6842	Food Preparation and Serving Related Occupations	0.0476	Farming, Fishing, and Forestry Occupations
Fasttext - SG	5	0.2264	0.1818	0.6842	Food Preparation and Serving Related Occupations	0.0149	Construction and Extraction Occupations
Fasttext - SG	7	0.2151	0.1710	0.5556	Legal Occupations	0.008	Production Occupations
Fasttext - SG	13	0.2125	0.1664	0.5556	Legal Occupations	0.0149	Production Occupations
Fasttext - CBOW	5	0.2095	0.1736	0.6316	Food Preparation and Serving Related	0.0149	Construction and Extraction Occupations

					Occupations		
Fasttext - CBOW	7	0.2052	0.1677	0.6316	Food Preparation and Serving Related Occupations	0	Construction and Extraction Occupations
Fasttext - CBOW	13	0.2055	0.1636	0.6316	Food Preparation and Serving Related Occupations	0.0298	Construction and Extraction Occupations