Proceedings of the Twenty-Ninth AAAI Conference on Innovative Applications (IAAI-17))

Large-Scale Occupational Skills Normalization for Online Recruitment

Faizan Javed, Phuong Hoang, Thomas Mahoney, Matt McNair
CareerBuilder
5550-A Peachtree Parkway
Peachtree Corners, GA 30092
{Faizan.Javed, Phuong.Hoang, Thomas.Mahoney, Matt.McNair}@CareerBuilder.com

Abstract

Job openings often go unfulfilled despite a surfeit of unemployed or underemployed workers. One of the main reasons for this is a mismatch between the skills required by employers and the skills that workers possess. This mismatch, also known as the skills gap, can pose socio-economic challenges for an economy. A first step in alleviating the skills gap is to accurately detect skills in human capital data such as resumes and job ads. Comprehensive and accurate detection of skills facilitates analysis of labor market dynamics. It also helps bridge the divide between supply and demand of labor by facilitating reskilling and workforce training programs. In this paper, we describe SKILL, a Named Entity Normalization (NEN) system for occupational skills. SKILL is composed of 1) A skills tagger which uses properties of semantic word vectors to recognize and normalize relevant skills, and 2) A skill entity sense disambiguation component which infers the correct meaning of an identified skill by leveraging Markov Chain Monte Carlo (MCMC) algorithms. Data-driven evaluation using end-user surveys demonstrates that SKILL achieves 90% precision and 73% recall for skills tagging. SKILL is currently used by various internal teams at CareerBuilder for big data workforce analytics, semantic search, job matching, and recommendations.

1 Introduction

Labor markets around the world are currently experiencing a perplexing quandary: high levels of unemployment or underemployment while job openings are at a record level. A recent report[1] from McKinsey discussed how employers in major European economies are facing a growing crisis of not finding people with the necessary skills to fill even entry level positions. At the same time, the European Union has 5.6 million young people without jobs. Similarly, in the U.S., a recent Labor department report[2] showed that there were a record number of job openings. While the U.S unemployment rate is less than 5%, there has also been an increase in the number of underemployed workers. Employers are not able to find qualified workers even for vocational jobs. While some employers have outsized expectations for even basic entry level jobs, the primary reason for persistently high levels of job openings is the skills gap. A recent study found that up to 80% of the engineers in India were unemployable[3] because engineering colleges were not teaching skills applicable in industry. The skills gap can pose socio-economic challenges for an economy: persistently high (youth) unemployment and loss in profits for companies and seriously hinder economic growth. To analyze and close the skills gap, it is imperative to have an automated system that can leverage a skills taxonomy to accurately detect skills in human capital data such as resumes and job ads. The resulting data forms the foundation for labor market analysis and consequently facilitates reskilling and workforce training programs. Automated skill systems can also be used in job matching and recommendation systems to better match candidates to jobs and reduce unemployment. Such skills systems also find application in compensation analytics which help quantify the value of specific skills and assist in improving employee wages.

There are various skills taxonomy and extraction systems. ESCO[4] is a European Commission project to categorize skills, occupations and other relevant competencies. It aims to provide semantic interoperability between labor markets and education and training programs. No information is available on what techniques and methodologies were used to create the ESCO taxonomies. The approach discussed in (Kivimäki et al. 2013) uses the LinkedIn skills taxonomy in conjunction with the spreading activation algorithm applied on the Wikipedia hyperlink graph to extract both inferred and explicitly stated skills from text. A skill inference model based on social graph connections is discussed in (Wang et al. 2014). This approach also uses data from LinkedIn and builds a factor graph model using textual information contained in the *Skills & Expertise* section, personal profile connections (shared majors, titles, companies and universities) and skill connections (skills that co-occur together). While the model based on skill connections is more accurate than the one that uses only profile connections, the joint model that uses both types of connections gives the best results.

[1]http://www.bbc.com/news/education-25714313

[2] http://money.cnn.com/2016/02/09/news/economy/america-5-6-million-record-job-openings/

[3]http://www.wsj.com/articles/indias-skills-shortfall challenges-modis-manufacturing-vision-1470653407

[4]https://ec.europa.eu/esco

The LinkedIn Skills system (Bastian et al. 2014) uses a data-driven approach to build a skills folksonomy. The folksonomy building pipeline consists of discovery, disambiguation and deduplication steps. The system also consists of a skills inference component which uses profile attributes such as company, title, and industry (among others) as features. The approach is similar to the skills inference model in (Wang et al. 2014) except that it uses a Naïve Bayes instead of a graph-based model. Skills recommendation and inference also finds application in talent management in large enterprises. Varshney et al. (Varshney et al. 2013) discuss a matrix factorization based approach to skill recommendation. This approach also leverages employee data from enterprise social networking tools, Human Resources (HR) and management data.

Our previous work (Zhao et al. 2015) gave an overview of an early version of SKILL, a system that utilized a novel approach to Named Entity Normalization (NEN) of occupational skills by leveraging properties of semantic word vectors. In this paper we build on that work and provide more details on the deployed SKILL system; more specifically we discuss the skills tagging algorithm as well as the skill entity sense disambiguation component. As the system has been in production for over a year, we also discuss i) the wide range of use cases at CareerBuilder[5] (CB), ii) end user feedback that resulted in improvements to the system, and iii) best practices and lessons learned from deploying and maintaining the system in production for a global customer base.

The rest of the paper is organized as follows. Section 2 reviews the SKILL system and briefly describe its components. Section 3 discuss the key improvements to the system. Next, we present how our application was designed, developed and deployed in sections 4 and 5. We then discuss its usage, payoff, and maintenance in sections 6 and 7. Finally, we conclude our work and suggest areas that require further study in section 8.

2 SKILL System Overview

Some key challenges of our tasks are summarized below. An effective skill system should be able to do:

1. Recognize skill entity from both job postings and resumes. These sources are semi-structured and may contain varying degrees of noises.
2. Handle name variations. The skill entity *Artificial Intelligence* can be in plural form *Artificial Intelligences*, it can also be in acronym form *AI* or contain typos *Artificially-Intelligent*.
3. Leverage semantic context to recognize unspecified skill entities. A Statistician job posting contains *correlation analysis, multivariate regression* skill entities as well as some other terms such as (requires) *PhD* or *masters* but does not contain *logistic regression* and *hypothesis testing*. These unspecified skills should be recognized with reasonable confidence levels.
4. Reduce false positives in tagging skills with multiple senses (e.g., *Server* has *food serving* and *information technology* senses).

Some of these challenges (1 and 3) are unique in the recruitment domain while other challenges (2 and 4) also exist in typical NEN task. In this section, we describe the SKILL system which aims to address all these challenges.

The architecture of our system is shown in Figure 1. On the left of Figure 1 we present the skill taxonomy generation. Once this task is completed, we employ the resulting *Skill Library* for recognition task, as shown in details on the right of Figure 1. For additional details for each of the listed components of the system, please refer to our previous work (Zhao et al. 2015). In this paper, we want to elaborate more on the improvements to the existing system.

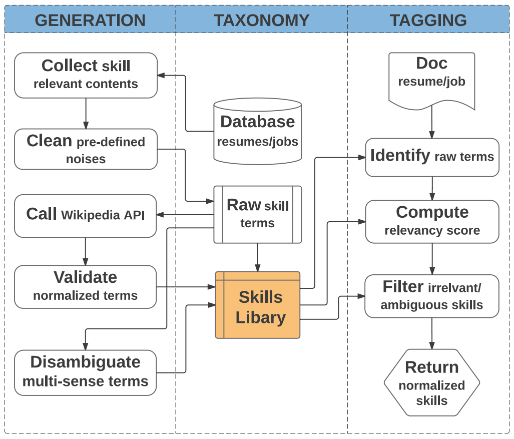


Figure 1: Architecture of the SKILL system.

2.1 Skill Taxonomy Generation

Collect In order to generate candidate skills, we collect skill related contents from over 60 million candidate resumes and 1.6 million job postings available at CB online career site. The selected section can be Skills, Technical Skills, Technical Proficiency for resume or the Requirements section in job posting. We do not infer the content or meaning of the extracted content as the goal of this step is to capture as much skill data as possible.

Clean We split text by punctuation, then remove noise, if any. The pre-defined noise dictionary contains of stopwords[6], country and city names, adverbs, adjectives and other predefined terms by domain expertise. Our goal here is to discard the extremely common words which contribute little to none semantic values in building the skill taxonomy.

Call After gathering raw terms (or seed phrases), we call Wikipedia API[7] for normalization and de-duplication. We

[5]http://www.careerbuilder.com/

[6] http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

[7]http://en.wikipedia.org/w/api.php

first do an *open search* action using seed phrases as the input query followed by a *query* action for associated Wikipedia documents, if any, and collect *category* tags and *redirections*.

Validate The goal of this step is to retain seed phrases directly related to occupational skills. We rely on keywords from the Standard Occupational Classification (SOC) system[8] to validate returned Wikipedia *category* tags hence make a decision on the qualification of the seed skills. Overall, an input query will be considered a skill *surface form* if its resulting Wikipedia document title *category* tags pass the SOC keyword screening.

Disambiguate The objective here is to address the Word Sense Disambiguation (WSD) problem. For example, a seed skill phrase is linked to multiple qualified Wikipedia documents, hence multiple normalized skills. Our initial approach for WSD utilized the Google Search API[9]. For instance, given a skill phrase with multiple sense, we select the one with highest Google Search ranking (by relevancy).This approach, however, shows the obvious weakness for not considering the semantic context that skill belongs to, leading us to develop a more robust approach toward the WSD task, as described in detail in section 3.1.

Skill Library In our current taxonomy, there are 39K surface forms that are mapped to 26K normalized skills entities. Each skill entity contains a unique identification code (skill ID), its raw term (or surface form), its normalized term, its vector of related surface forms, the corresponding vector of the cosine similarities and a skill type (e.g., *hard skill*, *soft skill*, and *certification*). See Table 1 for illustration of a typical skill entity in our skill library.

Table 1: Example of a skill entity the current skill library. Note that full sizes of the related surface forms vector and the cosine similarities are 200.

Skill ID	KS4401T642KKKL4FQJMF
Raw Term	restaurant
Normalized Term	Restaurant Operation
RelatedSF Vector	{hospitality, food services, customer service, clean room, retail stores, public utilities, vacuum, . . . }
Cosine Vector	{0.774, 0.727, 0.719, 0.709, 0.707, 0.668, 0.661, . . . }
Skill Type	Hard Skill

2.2 Skill Tagging

Identify We identify seed skills from a given input document by direct matching with the taxonomy. We break the input text into unigram tokens, assemble n-grams sequentially then match against the existing taxonomy which is stored as a hashmap for keys (surface forms).

[8]http://www.bls.gov/soc/major_group.htm

[9]https://developers.google.com/web-search/docs

Compute We compute the relevancy score for each matching surface form. For instance, given a target surface form, its relevancy score accounts for the percentage of surface forms from its *word2vec* vector of related surface forms out of all matching surface forms in the input document. It is important to note that the initial scoring method weight all surface forms the same regardless of their relatednesses to the target surface form. The improvement to the relevancy scoring method resolves this shortcoming by weighting each related surface form proportionally to its *cosine similarity*. See Section 3.2 for more details.

Filter Any surface form with a normalized relevancy score higher or equal than 2%, equivalent to 70% normalized score, will be recognized. This threshold value is selected empirically and justified by domain expertise on cross-SOC sample resumes. Normally, given a matching surface form we only expect one normalized skill entity in the skill library to be tagged. In many cases, an ambiguous surface form causes multiple senses to be tagged. We propose a relativity based thresholding approach that penalizes this co-occurrence, resulting in a more accurate tagging for ambiguous skills.

Table 2: A job posting sample.

Job Description We are currently seeking a <u>financial</u> accountant for the Honolulu area. The ideal candidate would collect and analyze <u>financial</u> data to assist in making <u>financial</u> decisions. **Financial Accountant Job Duties:**

- Collect, analyze, and summarize data and trends.
- Prepare monthly, quarterly, and annual statements based of reporting.
- Provide <u>financial</u> advice while complying with federal ANS state laws.
- Must have current knowledge of <u>financial</u> regulations.
- Monitoring <u>budgets</u>, developing <u>forecasts</u>, and investigating <u>variances</u>.

Job Requirements:

- Knowledge of SFAS rules.
- Proficient on <u>Excel</u>.
- <u>CPA</u> and/or <u>forensic</u> <u>accounting</u> experience preferred.
- Must be detail oriented and business knowledge

Pay: $65K-$85K, depending on experience

Return After surface forms are successfully extracted from the input document and validated against the skill library, the skill service returns an array of extracted skill entities. Each returned skill entity contains a unique identification code, its raw term (or surface form), its normalized term, its confidence score (or relevancy score), and its type. An example of the skill tagging service for a job posting[10] is illustrated in Table 2 and 3.

Table 3: List of tagged skills of job sample (Table 2).

Raw Term	Normalized Term	Relevancy Score	Type
accounting	Accounting	.95	Hard Skill
financial	Finance	.92	Hard Skill
forensic accounting	Forensic Accounting	.90	Hard Skill
budgets	Budgeting	.87	Hard Skill
CPA	Certifiedt Public Accountant	.87	Certification
forecasts	Forecasting	.85	Hard Skill
Excel	Microsoft Excel	.85	Hard Skill

3 Improvements of SKILL system

3.1 Word Sense Disambiguation (WSD)

In this section, we briefly discuss the improved version of WSD as introduced in section 2.1. Inspired by the previous works of Singh, Wick, and McCallum (2012) and Singh et al. (2011), we adopt the Metropolis-Hastings algorithm, a Monte Carlo Markov Chain (MCMC) method, to build Macau, a large scale skill sense disambiguation system in the online recruitment domain (Luo et al. 2015).

Macau's primary objective is to assign the most appropriate skill sense to a multi-sense skill entity with respect to a given context. A skill entity is defined and identified by our taxonomy and tagging system (Zhao et al. 2015). We employ the *word2vec* tool (Mikolov et al. 2013) to obtain a vector representation of the context of a skill entity. These vectors are then used as input for clustering such that in each cluster, the aggregated contexts (represented by vectors) can be used to determine a skill sense. Figure 2 shows an example of two clusters for STOCKS that clearly represent two distinct senses, e.g., *Warehouse Inventory* and *Finance*.

The evaluation of the new WSD system was performed through a data-driven approach. We randomly selected more than 29K resumes, covering 90% of all ambiguous skills (suggested by the Macau system) across all industries categorized by SOC. For each ambiguous skill, around 400 resumes were selected for skill tagging validation by three human evaluators. Overall, the Macau system attains 84% in tagging precision with respect to the input text. It is also worth mentioning that the number of different senses for a given skill entity (number of clusters) does not need to be predefined in our proposed method. This leads to a major advantage over other clustering algorithms such as k-means and LDA. For large-scale datasets, we also propose a distributed system based on MCMC clustering algorithm to parallelize the clustering process.

[10]http://www.careerbuilder.com/job/JHS6H0629QTZ097T2FQ

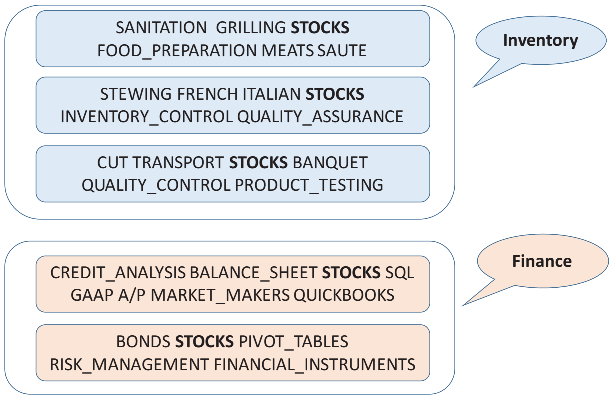

SANITATION GRILLING STOCKS FOOD_PREPARATION MEATS SAUTE
STEWING FRENCH ITALIAN STOCKS INVENTORY_CONTROL QUALITY_ASSURANCE
CUT TRANSPORT STOCKS BANQUET QUALITY_CONTROL PRODUCT_TESTING
Inventory
CREDIT_ANALYSIS BALANCE_SHEET STOCKS SQL GAAP A/P MARKET_MAKERS QUICKBOOKS
BONDS STOCKS PIVOT_TABLES RISK_MANAGEMENT FINANCIAL_INSTRUMENTS
Finance

Figure 2: Sample documents representing the two senses of the STOCKS skill term. Each colored box represents a document that containing a set of skills extracted from the same resume or job posting

3.2 Skills Tagging

This section discusses the main improvements in the Skill tagging algorithm that was introduced in section 2.2.

Algorithm 1 takes as input the list of seed skill phrases $\langle x, w, \mathbf{v}, \mathbf{c}\rangle$ where x is a raw term (or surface form), w is its normalized term (or skill sense) from the taxonomy, $\mathbf{v}$, $\mathbf{c}$ is the vector of the related surface forms and the corresponding *word2vec* cosine distances. It returns a list of normalized skill objects $\langle x, w, \mathbf{v}, \mathbf{c}, \xi\rangle$ where only the most relevant skills are retained and ξ is the relevancy score with respect to the input content.

Algorithm 1 Skills Tagging

```
Input: SKILLS_raws: list of raw skill objects $\langle x, w, \mathbf{v}, \mathbf{c}\rangle$.
Output: SKILLS_norms: list of normalized skill objects $\langle x, w, \mathbf{v}, \mathbf{c}, \xi\rangle$
 1: for each $\langle x, w, \mathbf{v}, \mathbf{c}\rangle \in$ SKILLS_raws do
 2:     $\xi \Leftarrow$ getRelevancyScore($x, w, \mathbf{v}, \mathbf{c}, X$)        ▷ see Algorithm 2
 3:     if isMultiSensesSF($x$) then
 4:         $w' \Leftarrow$ filterWSD($x, W, \Xi$)        ▷ see Algorithm 3
 5:         SKILLS_norms $\Leftarrow \langle x, w', \mathbf{v}, \mathbf{c}, \xi\rangle$
 6:     else
 7:         SKILLS_norms $\Leftarrow \langle x, w, \mathbf{v}, \mathbf{c}, \xi\rangle$
 8:     end if
 9: end for
10: return SKILLS_norms
```

On line 2 of Algorithm 1, we compute relevancy score using function `getRelevancyScore` that is described in Algorithm 2. The raw scores were extremely low with 2% median and 0.003% variance, making relevancy ranking difficult. We utilized Beta distribution fitting to address this problem (line 3 of Algorithm 2). The parameters for the Beta distribution ($\alpha = 0.1627, \beta = 6.2385$) were empirically chosen so that the final relevancy scores span more evenly on the desired [0,1] interval. Only surface forms that score

Algorithm 2 getRelevancyScore: Relevancy score computation

```
Input: $(x, w, \mathbf{v}, \mathbf{c}, X)$ where $X$ is the set of all candidate
    surface forms from the input text.
Output: Confidence score $\xi$ of surface form $x$ with respect
    to the input context.
1: (Legacy relevancy score:
       $$\lambda(x) = \frac{\sum_{j;x_j \in X} I_{\mathbf{v}}(x_j)}{\sum_{j;x_j \in X} I_X(x_j)} \quad , \qquad (1)$$
   where $I_A(x)$ is the indicator function s.t. $I_A(x) = 1$ if
   $x \in A$, and 0 otherwise.)
2: New relevancy score:
       $$\xi(x) = \frac{\sum_{j;v_j \in X} c_j}{\sum_{j;v_j \in \mathbf{v}} c_j} \quad , \qquad (2)$$
   where $v_j$ and $c_j$ denote the $j$ component of $\mathbf{v}$ and $\mathbf{c}$
   respectively.
3: $\xi \Leftarrow$ fitBetaDist($\xi$)
4: if $\xi \geq \alpha = 70\%$ then
5:     return $\xi$
6: end if
```

Algorithm 3 filterWSD: Skills entity disambiguation

```
Input: $(x, W, \Xi)$ where $x$ is an ambiguous surface form
    multiple senses, $W = \{w_i\}$ denotes the set of all possi-
    ble senses with the set of relevancy scores $\Xi = \{\xi_i\}$.
Output: List of strongest senses $\{w_k\}$ that can be referred
    to by the given ambiguous surface form $x$ based on rel-
    evancy scores.
1: return $w_{max}$ s.t. $\xi_{max} = \max\{\xi_i\}_{i=1}$
2: for each sense $w_i \in W \setminus w_{max}$ do
3:     if $\frac{\xi_i}{\xi_{max}} \geq 90\%$ then
4:         return $w_i$
5:     end if
6: end for
```

70% or higher are returned (line 4 of Algorithm 2).

Algorithm 1 continues on line 3, where function `isMultiSensesSF` examines the ambiguity of the input surface form by validating it against the WSD list from the *skill library*. If multiple skill senses are recognized, we use function `filterWSD`, on line 4 of Algorithm 1, to select the most appropriate senses. Here, the skill sense with the highest relevancy score is returned together with other senses that are at least 90% as good, relatively (line 1-3 of Algorithm 3). Finally, on line 10 of Algorithm 1 we obtain the set of normalized skills.

The key improvement of relevancy scoring is described in Algorithm 2. Given a surface form matched from an input text, our initial attempt, as defined in equation (1), is simply taking the ratio of the size (surface forms count) of the intersection of $\mathbf{v}$ and X to the size of X . It is important to note that all surface forms in the *word2vec* vector of related surface forms $\mathbf{v}$ are treated as equally important. This leads to a major drawback. For example, if a resume contains both *C++* and *Visual C++* then the relevancy score of *C (Programming Language)* should be high; while if *Hewlett-Packard Graphics Language (HPGL)* and *Automata Theory* appear instead, then the score should be lower. Unfortunately, there is no distinction in relevancy score among these two cases under the frequency-based approach. To address this drawback, we propose a weighted semantic relevancy scoring approach. As described in equation (2), the new relevancy score takes into account the weight of each matched surface forms in the vector of related surface forms. These weights are in fact the *cosine similarities* of *word2vec* vector of the related skills. Hence, for a given matched surface form, the occurrence of its closely related surface forms increases its relevancy score substantially while the occurrence of its loosely related surface forms does not impact its relevancy score at much.

The evaluation of the current skill tagging framework was performed similarly to our previous work (Zhao et al. 2015) through sampling based users survey. While an automatic approach is available, we believe that the users are the one know best about their skills. We analyzed over 1,300 responses of active users (6 months) across all industries categorized by SOC . To measure precision, we request the users to validate the top 10 skills per resume ranked by relevancy scores. To measure recall, we ask the user to add up to 5 skills that are missing from the presented list.

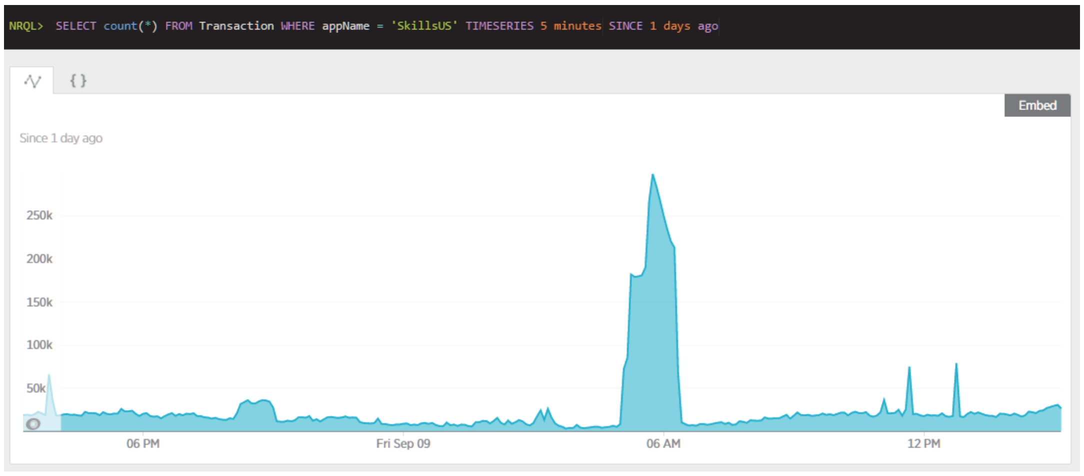
Table 4: Skill tagging comparison between versions: Precision and Recall.

Version	**Precision**	**Recall**
Old	82%	70%
Current	**90%**	**73%**

Table 5: Skill tagging results per confidence score level. Correlation between confidence score and approval rate is 0.81.

Relevancy Score	**Approved Skills**	**Total Skills**	**Approval Rate**
.95	130	149	.8725
.90	316	371	.8518
.85	455	546	.8333
.80	317	407	.7897
.75	109	146	.7466
.70	8	12	.6667

The results show that the current skill tagging framework attains 90% precision and 73% recall which is better than 82% precision and 70% recall of the old version (Zhao et al. 2015). Moreover, a strong correlation between the relevancy score and user approval rate is observed. Table 5 clearly shows that the higher the relevancy score, the higher the chance of approval by users.



```
NRQL> SELECT count(*) FROM Transaction WHERE appName = 'SkillsUS' TIMESERIES 5 minutes SINCE 1 days ago
```

Figure 3: One day's traffic to the SKILL service. The x-axis is time and the y-axis is the number of requests per five minutes. This is representative of what we currently typically handle. These spikes can be even more sudden and often led to elevated error rates while our auto-scaling caught up with the increased request volume.

4 Technical Design Overview

Our projects require a successful synergy between data scientists and data engineers to move from prototype to production. The SKILL system is one of the first projects at CB that demonstrated successful collaboration between the two organizational teams. Projects with a strong data science component initiate with the data scientists conducting R&D research spikes to build prototypes to verify the feasibility of business ideas. The data engineers are involved in technical design discussions as soon as it becomes evident that the prototype can move to production. The synergy between the two teams is critical because it is important to understand the limitations of open-source tools used by data scientists in production environments. These limitations sometimes also influence the tools used by the data scientists but in general, we don't place hard restrictions during the prototyping phase.

For the SKILL system, the goal was to provide a skill tagging service as a micro service to both internal and external customers. The core R&D components of the SKILL system (see Figure 1) are similar in design to previous NEN efforts as well as systems that generate taxonomies from data sources using knowledge bases such as Wikipedia. Both the taxonomy generation and WSD phases are off-line processes that can be scheduled to run on demand. The taxonomy generation phase makes extensive use of data scraping, cleaning and extraction scripts that run millions of job postings. Since disambiguation is a computationally intensive process that usually involves clustering, it was also designed as an offline batch job. The skills tagging algorithm was developed to support the near real-time requirements of a web service.

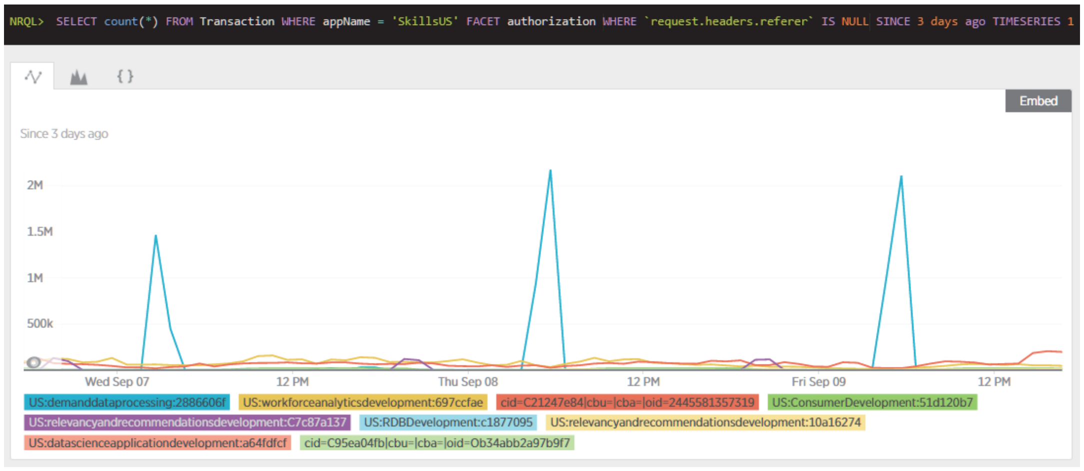
On the data engineering side, the SKILL service was developed as a Java 5 web service using the servlets framework. Library file generation is done offline ahead of time through a separate process. These libraries are loaded into memory at web service startup time. The service is RESTful; it holds no state and it always returns the same response for a given request. The service accepts HTTP GET and POST requests and handles the two identically; POST support is provided exclusively as a means of accepting larger payloads (most of our customers use POST as the default for all skills extraction requests).

An incoming request must contain a content string containing the text upon which skills extraction should be performed. A request may also optionally provide a language string (we support skills tagging in 22 languages); a threshold decimal value between 0 and 1 for controlling minimum relevancy; and an *auto_thres* Boolean value which controls the extractor's behavior on inputs containing 150 or fewer words. After successful extraction, the service returns a JSON payload containing an array of extracted skill objects. Each skill object contains a unique identifier code, its normalized term, a confidence score between 0 and 1, and a skill type.

5 Development and Deployment

The core system was developed by two data scientists over a period of one year. The taxonomy component was developed in R, Hadoop and C (for the *word2vec* word vectors), while the WSD component was developed in C++ to alleviate the computational cost of large-scale clustering processes. Since the skills tagging algorithm is a core component of the deployed SKILL service, it was implemented in Java to take advantage of CB's deployment and scaling expertise.

Over time, the technical implementation of the service has evolved. The initial implementation was not written with clean code principles in mind; over time, functions have been shortened, new functions and classes have emerged, redundancies have been eliminated, and variables, functions, and classes have all been renamed for clarity. The codebase was upgraded to Java 8 and uses features new in the latest version of the Java SDK, such as Optionals and the Streams



```
NRQL> SELECT count(*) FROM Transaction WHERE appName = 'SkillsUS' FACET authorization WHERE `request.headers.referer` IS NULL SINCE 3 days ago TIMESERIES 1
```

Figure 4: Number of Requests per Client ID over the last three days. Demand Data Processing is clearly the biggest consumer of Skills. Note that every caller represented here is sending requests on the order of thousands per hour at least.

API, for more expressive and maintainable code.

The development team that deploys and operates the Skills web service was also able to reduce lines of codes in the project by ripping out the old servlet configuration code and using a homegrown web service chassis instead. This chassis is used in many web services at CareerBuilder and handles application startup, request and response de/serialization, and other boilerplate web service functionalities, thereby removing complexity from the SKILL service and improving ease of maintenance. The service has also been enhanced to respond with a variety of descriptive HTTP status codes for various errors, such as *400 Bad Request* errors for improperly structured requests and *401 Unauthorized* errors for requests that do not present the required authentication credentials.

Operating the skills extraction service at production scale has presented some interesting challenges. The service is deployed and operated as an "always-on" HTTP endpoint, but we have dealt with a significant learning curve in ensuring its availability through sudden traffic bursts. Unlike some more generalized web services such as our resume parsing and geocoding services, the skills extraction service currently is rarely used in user-facing code paths on the site; rather, the vast majority of its traffic comes from asynchronous batch processes that other CareerBuilder teams use in building recommendations, enriching candidate data, and so on, as seen in Figure 4. Many of these batch processes run at a very high level of concurrency and ramp up quite quickly once initiated, which results in extremely spiky traffic to our service. Figure 3 shows a typical request volume spike from one-day traffic.

Scaling up our server fleet to handle these traffic spikes smoothly proved quite difficult. Our first solution to this was to simply run more instances at all times, but this was wasteful and expensive. The service itself was already quite optimized, so there were no easy gains to be made with regards to performance. Ultimately we found that the best solution was to consult with our users and ask them to build gradual scaling into their batch processes. Currently, a locally-deployable, offline version of the skills extractor is being developed that will enable teams to perform skills enrichment without sending requests to our service at all, at whatever speed their own hardware will allow.

6 Usage and Payoff

The skills extraction service provides a variety of benefits in the CareerBuilder ecosystem. For example, our candidate enrichment system parses skills from uploaded resumes, which are then used for generating job recommendations. EMSI (Economic Modeling Specialists Intl.), our partner company, leverages skills extraction in multiple areas of their business:

> *EMSI's mission is to use data to connect and inform people, employers, and educators. Skills are at the core of these relationships, so being able to take many unstructured data sources and connect them with a common skills vocabulary is crucial. Skills normalization is enabling us to make great data products that bring together the labor market demand side (job openings), the supply side (resumes and social media profiles), and the often-ignored side of human capital development (education and training offerings).*

We also use skills extraction in our semantic search API to correlate keywords and/or documents. Strong search functionality is a critical business differentiator for CB, and the skills extraction engine significantly improves our ability to understand semantic queries in applications dealing with human capital data. For example, CB's Resume Database product allows recruiters to search millions of job seeker resumes; with our semantic search engine powered by the SKILL service, we are able to recognize known skills in recruiters' search queries, and expand these searches to include highly related skills (Shalaby et al. 2016; AlJadda et al. 2014). The SKILL service is also a crucial component of CB's recommendations engine, which serves millions of job recommendations to job seekers every week. We are also experimenting with using the SKILL service to aid in correlating skill and compensation data in our Compensation Portal B2B product. The SKILL service is used in some form or another by nearly every product team at CB and acts as a strong business-differentiating tool that furthers our overall mission of connecting candidates with jobs.

7 Maintenance

The Skills taxonomies and extraction algorithms are still being improved upon today. Occasionally a new version of either the taxonomy libraries or the extraction logic (or both) will be made available in production. We use the *major.minor.patch* versioning scheme and maintain no more than two live major versions at a time. When a new version is released, the oldest live version is scheduled for retirement, and we communicate a *sunset date* to our customers.

We occasionally develop small incremental changes to our taxonomies that must be released as separate versions (to preserve data integrity and our versioning contracts) but consist of 90% or higher identical data from the previous version. To accomplish this effectively while minimizing overhead, we developed a small library that implements persistent collections, allowing us to store e.g., versions 4.0, 4.1, and 4.2 libraries without incurring 3 times the memory cost.

Large scale adoption of the service has resulted in many improvements and enhancements to the system over time. WSD was added to the service in version 2.0 after users requested a more robust skills disambiguation mechanism. Continuous feedback loops from users also helped in removing noise from the taxonomy. Skills categorization was added to the taxonomy after users suggested that there was a market need for it in several products. On the user-facing side, we are also working to give our users the option to see the skills extracted from their resume the instant they upload their resume on the site. This feature will also allow the users to make corrections or additions to their list of skills, and we will be able to track these edits to further improve our understanding of how our users reason about skills.

8 Conclusion and Future Work

The skills gap is increasingly cited as the main reason for the disparity in a large number of job openings and underemployment in economies around the world. We argue that accurately detecting skills in human capital data is a first step to resolving this socio-economic problem. To this end, we describe the SKILL system for skill normalization that has been in production at CareerBuilder for more than a year. More specifically we follow up on our previous work, which was back then an emerging prototype, by describing how the system evolved over time as it gained greater traction and usage across the company. We also focus on the collaboration between the data science and data engineering teams and describe how both organizational teams are needed to bring large scale, high impact ideas to fruition.

We have received extensive and valuable feedback from our internal stakeholders as well as external customers on areas for improvement and are currently researching several future directions for the system. We plan to improve the system by supporting case-sensitive tagging to minimize false positives and build a more comprehensive skills hierarchy. As we expand the system in various international markets we will also support multilingual skills tagging and taxonomies. To support our compensation analytics and career path efforts, we plan to extend the system with skills proficiency, expertise inference and skill effort capabilities.

References

AlJadda, K.; Korayem, M.; Grainger, T.; and Russell, C. 2014. Crowdsourced query augmentation through semantic discovery of domain-specific jargon. In *Big Data (Big Data), 2014 IEEE International Conference on*, 808–815. IEEE.

Bastian, M.; Hayes, M.; Vaughan, W.; Shah, S.; Skomoroch, P.; Kim, H.; Uryasev, S.; and Lloyd, C. 2014. Linkedin skills: large-scale topic extraction and inference. In *Proceedings of the 8th ACM Conference on Recommender systems*, 1–8. ACM.

Kivimäki, I.; Panchenko, A.; Dessy, A.; Verdegem, D.; Francq, P.; Fairon, C.; Bersini, H.; and Saerens, M. 2013. A graph-based approach to skill extraction from text. *Graph-Based Methods for Natural Language Processing* 79.

Luo, Q.; Zhao, M.; Javed, F.; and Jacob, F. 2015. Macau: Large-scale skill sense disambiguation in the online recruitment domain. In *Big Data (Big Data), 2015 IEEE International Conference on*, 1324–1329. IEEE.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Shalaby, W.; Jadda, K. A.; Korayem, M.; and Grainger, T. 2016. Entity type recognition using an ensemble of distributional semantic models to enhance query understanding. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, 631–636. IEEE.

Singh, S.; Subramanya, A.; Pereira, F.; and McCallum, A. 2011. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 793–803. Association for Computational Linguistics.

Singh, S.; Wick, M.; and McCallum, A. 2012. Monte carlo mcmc: Efficient inference by approximate sampling. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1104–1113. Association for Computational Linguistics.

Varshney, K. R.; Wang, J.; Mojsilovic, A.; Fang, D.; and Bauer, J. H. 2013. Predicting and recommending skills in the social enterprise. In *Proc. AAAI ICWSM Workshop Social Comput. Workforce*, volume 2, 20–23.

Wang, Z.; Li, S.; Shi, H.; and Zhou, G. 2014. Skill inference with personal and skill connections. In *COLING*, 520–529.

Zhao, M.; Javed, F.; Jacob, F.; and McNair, M. 2015. Skill: A system for skill identification and normalization. In *AAAI*, 4012–4018.