Large-Scale Occupational Skills Normalization for Online Recruitment

Phuong Hoang, Thomas Mahoney, Faizan Javed, Matt McNair

■ *Job openings often go unfilled despite a surfeit of unemployed or underemployed workers. One of the main reasons for this disparity 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 socioeconomic 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 article, 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. We discuss the technical design and the synergy between data science and engineering that was required to transform the system from a research prototype to a production service that serves customers from across the organization. We also discuss establishing customer feedback loops, which led to improvements to the system over time. SKILL is currently used by various internal teams at CareerBuilder for big data workforce analytics, semantic search, job matching, and recommendations.*

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. Although the U.S unemployment rate is less than 5 percent, there has also been an increase in the number of underemployed workers. While some employers have outsized expectations for 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 percent of the engineers in India were unemployable[3] because engineering colleges were not teaching skills applicable in the industry. The skills gap can pose socioeconomic challenges for an economy: persistently high (youth) unemployment and loss in profits for companies can 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 skill systems also find application in compensation analytics, which helps 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 by Wang et al. (2014). This approach also uses data from LinkedIn and builds a factor graph model using textual information contained in the skills and expertise section, personal profile connections (shared majors, titles, companies, and universities), and skill connections (skills that cooccur 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 connection types gives the best results.

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 skill inference model presented by Wang et al. (2014) except that it uses a Näıve Bayes instead of a graph-based model. Skill recommendation and inference also find application in talent management in large enterprises. 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 article, we build on that work and provide more details on the deployed SKILL system; more specifically, we discuss the skill 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 the wide range of use cases at CareerBuilder[5] (CB), end-user feedback that resulted in improvements to the system, and best practices and lessons learned from deploying and maintaining the system in production for a global customer base.

SKILL System Overview

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

1. Recognize skill entities from both job postings and resumes. These sources are semistructured and may contain varying degrees of noise.

2. Handle name variations. The skill entity artificial intelligence can be in plural-form artificial intelligences, it can also be in acronym-form AI, and it might contain typos artificially intelligent.

3. Leverage semantic context to recognize unspecified skill entities. A statistician job posting, for example, might contain correlation analysis and multivariate regression skill entities, as well as other skills required by PhD- or masters-level work, but not logistic regression and hypothesis testing. These unspecified skills should be recognized with reasonable confidence.

4. Reduce false positives in tagging skills with multiple senses. The term “stock,” for example, has meaning both in the context of food preparation and the context of finance.

Some of these challenges (1 and 3) are unique in the recruitment domain, while others (2 and 4) also exist in other typical NEN tasks. In this section, we describe the workings of the SKILL system, which aims to address the sum total of these challenges. Figure 1 summarizes our system architecture. On the left, we present the skill taxonomy generation. Once this task is completed, we employ the resulting skill library for the tagging task, as shown on the right of the figure.

Skill Taxonomy Generation

Collect To generate candidate skills, we collect skill-related contents from over 60 million candidate resumes and 1.6 million job postings available at the CB online career site. The selected section can be *skills, technical skills, technical proficiency* for resumes or the *requirements* section in job postings. We do not infer the content or meaning of the extracted content at this point, as the goal of this step is to capture as much skill data as possible.

Clean. We split text by punctuation, then remove any noise. The predefined noise dictionary contains stop words,[6] country and city names, additional adverbs and adjectives, and other predefined terms by domain expertise. Our goal here is to discard the ubiquitous words that contribute little to no semantic value in building the skill taxonomy.

Call. After gathering raw terms (also known as *surface forms*), we call the Wikipedia API[7] for normalization and deduplication. We do an open search action using seed phrases as the input query, followed by a query action for associated Wikipedia documents, if any, and then collect category tags and redirections.

Validate. The goal of this step is to retain surface form directly related to occupational skills. We rely on keywords from the standard occupational classification (SOC) system[8] to validate the returned

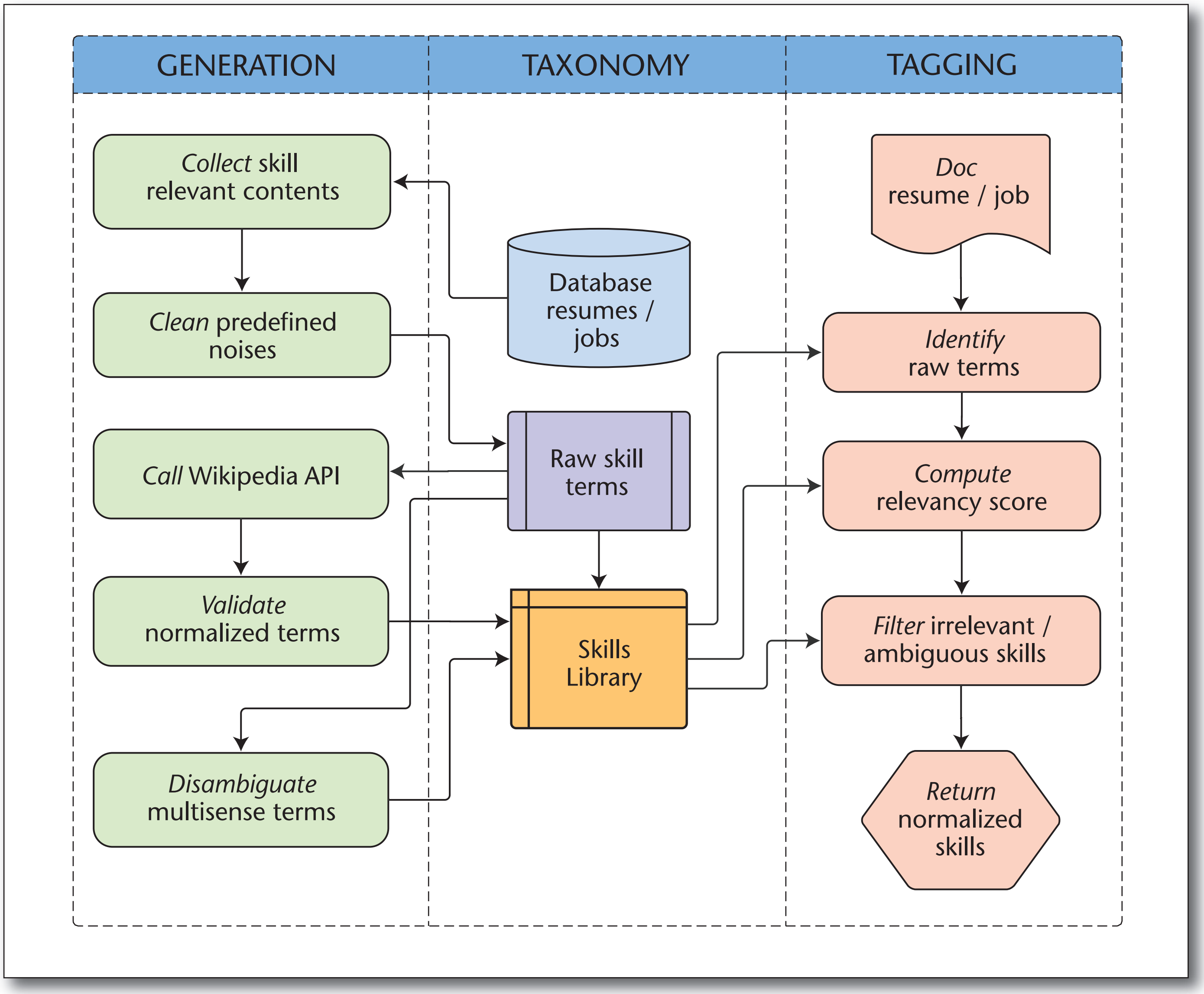
GENERATION
TAXONOMY
TAGGING
Collect skill relevant contents
Clean predefined noises
Call Wikipedia API
Validate normalized terms
Disambiguate multisense terms
Database resumes / jobs
Raw skill terms
Skills Library
Doc resume / job
Identify raw terms
Compute relevancy score
Filter irrelevant / ambiguous skills
Return normalized skills

Figure 1. Architecture of the SKILL System.

Wikipedia category tags and make a decision on the qualification of the surface form as 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 surface form links to multiple qualified Wikipedia documents, hence multiple normalized skills. Our initial approach for WSD utilized the Google Search API.[9] For instance, given a surface form with multiple senses, we select the one with the highest Google Search ranking (by relevancy). This approach, however, shows the obvious weakness in not considering semantic context, leading us to develop a more robust approach for the WSD task, as described in detail in the next section.

Skill Library. In our current taxonomy, there are 39,000 surface forms mapped to 26,000 normalized skill 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 (such as hard skill, soft skill, and certification). See table 1 for an illustration of a typical skill entity in our skill library.

Skill Tagging

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

Skill ID	KS1218C6MP9RN7WXMM37
Raw Term	business analysis
Normalized Term	Business Analytics
RelatedSF Vector	{business performance management, predictive analytics, data analytics, customer segmentation, technology strategy, business process mapping, value chain analysis, . . . }
Cosine Vector	{0.689818, 0.673068, 0.665908, 0.657712, 0.652762, 0.650666, 0.650395, . . . }
Skill Type	Hard Skill

Table 1. Example of a Skill Entity of the Skill Library.

Note that full sizes of the related surface forms vector and the cosine similarities are 200.

Compute. We compute the relevancy score for each matching surface form. Given a target surface form, its relevancy score is 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 weighed 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.

Filter. Any surface form with a normalized relevancy score higher than or equal to 70 percent 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 expect only 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.

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 resume sample is illustrated in tables 2 and 3.

Improvements in the SKILL system

Inspired by the previous works of Singh, Wick, and McCallum (2012) and Singh et al. (2011), we adopted 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 NLP that represent two distinct senses, such as *natural language processing* and *neural linguistic program*.

We evaluated the new WSD system using a data-driven approach. We randomly selected more than 29,000 resumes, covering 90 percent of all ambiguous skills (as 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 percent 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 (that is, the 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 data sets, we also propose a distributed system based on MCMC clustering algorithm to parallelize the clustering process.

Skill Tagging Relevancy

The key improvement of the skill tagging process lies in the relevancy score. In our initial attempt, the relevancy score was defined as the ratio of coexisting (from the input text) surface forms in the vector of the word2vec-related surface form out of all coexisting surface forms matched (Zhao et al. 2015).

Formally, let X be a set of all candidate surface forms extracted from an input text. Then for any surface form $x \in X$ and its word2vec vector v, the relevancy score is defined as:

$$RelScore(x) = \frac{\sum_{j;x_j \in X} I_X\left(x_j\right)}{\sum_{j;x_j \in X} I_v\left(x_j\right)} \tag{1}$$

where $I_A(x)$ is the indicator function s.t. $I_A(x) = 1$ if $x \in A$, and 0 otherwise. Note that we control the vector settings such that $x \notin v$, that is, a vector is not inclusive of its surface form.

It is important to note that all surface forms in the word2vec vector of related surface forms 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 (a related 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 between 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, this relevancy score takes into account the weight of each matched surface form in the vector of related surface forms. These weights are in fact the cosine similarities of the word2vec vector of related skills. Hence, if c denotes the vector of cosine similarities $c_j \in$ c, the new relevancy score is formally defined as:

$$RelScore(x) = \frac{\sum_{j;v_j \in X} c_j}{\sum_{j;v_j \in \mathrm{v}} c_j} \tag{2}$$

where v_j and c_j denote the $_j$ component of v and c respectively.

Therefore, for a 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 much.

In practice, the relevancy scores were extremely low with a 2 percent median and a 0.003 percent variance, making relevancy ranking difficult. This is because the size of the related surface forms is much larger than the size of the coexisting surface forms from an input text. We utilized beta distribution fitting to address this problem, as highlighted in figure 3. The distribution of the raw scores (depicted by black curve) is heavily right skewed (centered at 0.02), giving an incorrect perspective of low relevancy. After scaling through beta distribution, the final scores (depicted by red curve) span more evenly in the [0, 1] interval, making relevancy rating easier.

EDUCATION

Georgia Institute of Technology, Atlanta, GA
M.S. Computer Science (Specialization: Machine Learning). Aug 2015 - May 2018

Georgia Institute of Technology, Atlanta, GA
B.S. Chemical and Biomolecular Engineering. Aug 2009 - Dec 2011

TECHNICAL

Skills: Java, Android, Python, Git, software development, Unix/Windows

Project: A System for Automated Testing of Android Apps
Design architecture of a crawler-based robot system to automate mobile app testing.
Integrate various testing framework (e.g. Appium, SL4A) APIs into system implementation.

EXPERIENCE

Developer Intern, AT&T, Atlanta, GA. June 2017 – Present
Evaluate U-verse data and apply machine learning techniques to predict potential failures.

Developer (Part-time), BlueFletch Mobile, Atlanta, GA. Aug 2016 - April 2017.
Developed client Android apps (implemented app features, utilized APIs to achieve various functionalities efficiently, refined user interface and user interface flows).

Android Application Developer, Academia Sinica, Taipei, Taiwan. Jan 2016 June 2016
Implemented/tested SmartHear 2.0 app functionalities, and published to Google Play store.

Issuing Services Consultant (Software Development), FIME, Taipei, Taiwan. Dec 2013 Jan 2016
Coordinated with clients to define/refine requirements for software development projects.

ADDITIONAL

Fluent in Mandarin Chinese, volunteer at Atlanta Shakespeare Tavern, member of Atlanta Android Developers group, member of Taiwanese Student Association

Table 2. A Resume Sample.

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 70 percent or higher are returned.

The evaluation of our skill tagging framework was performed through sampling-based user surveys. While an automatic approach is available, we believe that the users are the ones who know best what skills they possess. We analyzed more than 1,300 responses of active users over six months across all industries categorized by SOC. To measure precision, we asked

Raw Term	Normalized Term	Relevancy Score	Type
software development	Software Development	.95	Hard Skill
machine learning	Machine Learning	.95	Hard Skill
automated testing	Test Automation	.93	Hard Skill
Android	Android (Operating System)	.92	Hard Skill
java	Java (Programming Language)	.90	Hard Skill
python	Python (Programming Language)	.88	Hard Skill
mobile app	Mobile App	.87	Hard Skill
unix	Unix	.85	Hard Skill
api	Application Programming Interface	.85	Hard Skill
biomolecular engineering	Biomolecular Engineering	.84	Hard Skill
chemical engineering	Chemical Engineering	.84	Hard Skill
mandarin chinese	Mandarin Chinese Language	.82	Hard Skill

Table 3. List of Tagged Skills of the Resume Sample Presented in Table 2.

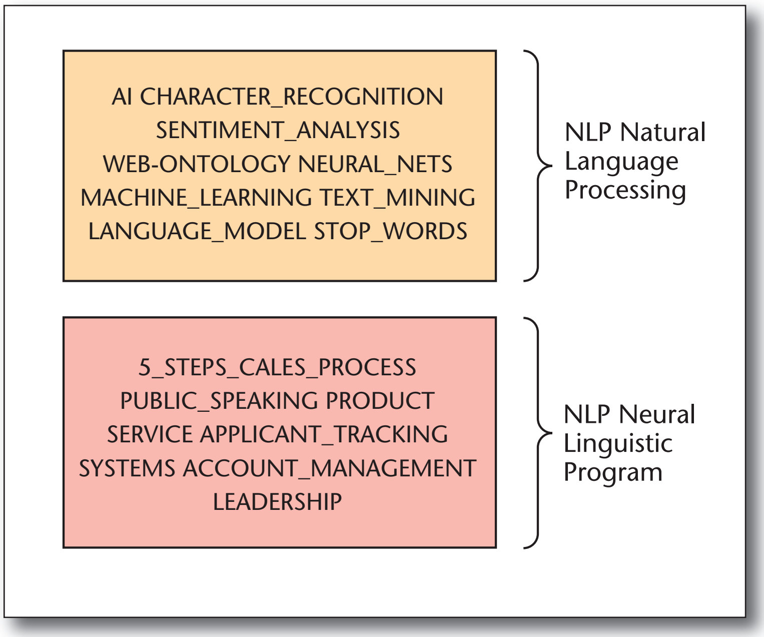
AI CHARACTER_RECOGNITION SENTIMENT_ANALYSIS WEB-ONTOLOGY NEURAL_NETS MACHINE_LEARNING TEXT_MINING LANGUAGE_MODEL STOP_WORDS
NLP Natural Language Processing
5_STEPS_CALES_PROCESS PUBLIC_SPEAKING PRODUCT SERVICE APPLICANT_TRACKING SYSTEMS ACCOUNT_MANAGEMENT LEADERSHIP
NLP Neural Linguistic Program

Figure 2. Sample Documents Representing the Two Senses of the NLP Skill Term.

Each colored box represents a set of related skill entities extracted from resumes or job postings.

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

Table 4. Skill Tagging Comparison Between Versions: Precision and Recall.

the users to validate the top 10 skills per resume ranked by relevancy scores. To measure recall, we ask the users to add up to five skills that were missing from the presented list.

The results show that the current skill tagging framework attains 90 percent precision and 73 percent recall, which is better than the 82 percent precision and 70 percent recall of the previous version (Zhao et al. 2015), as noted in Table 4. Moreover, a strong correlation between the relevancy score and the user approval rate is observed. Table 5 shows that the higher the relevancy score, the higher the chance of approval by users.

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 influence the tools used by the data scientists, but in general, we do not place hard restrictions during the prototyping phase.

For the SKILL system, the goal was to provide a skill tagging service (The SKILL service) as a microservice 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 to systems that generate taxonomies from data sources using knowledge bases such as Wikipedia. Both the taxonomy generation and WSD phases are offline 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 skill 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 skill extraction requests).

An incoming request must contain a content string with the text upon which skill extraction should be performed. A request may also optionally provide a language string (we support skill tagging in 22 languages), a threshold decimal value between 0 and 1 for controlling minimum relevancy, and an *auto_thres* Boolean value that 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.

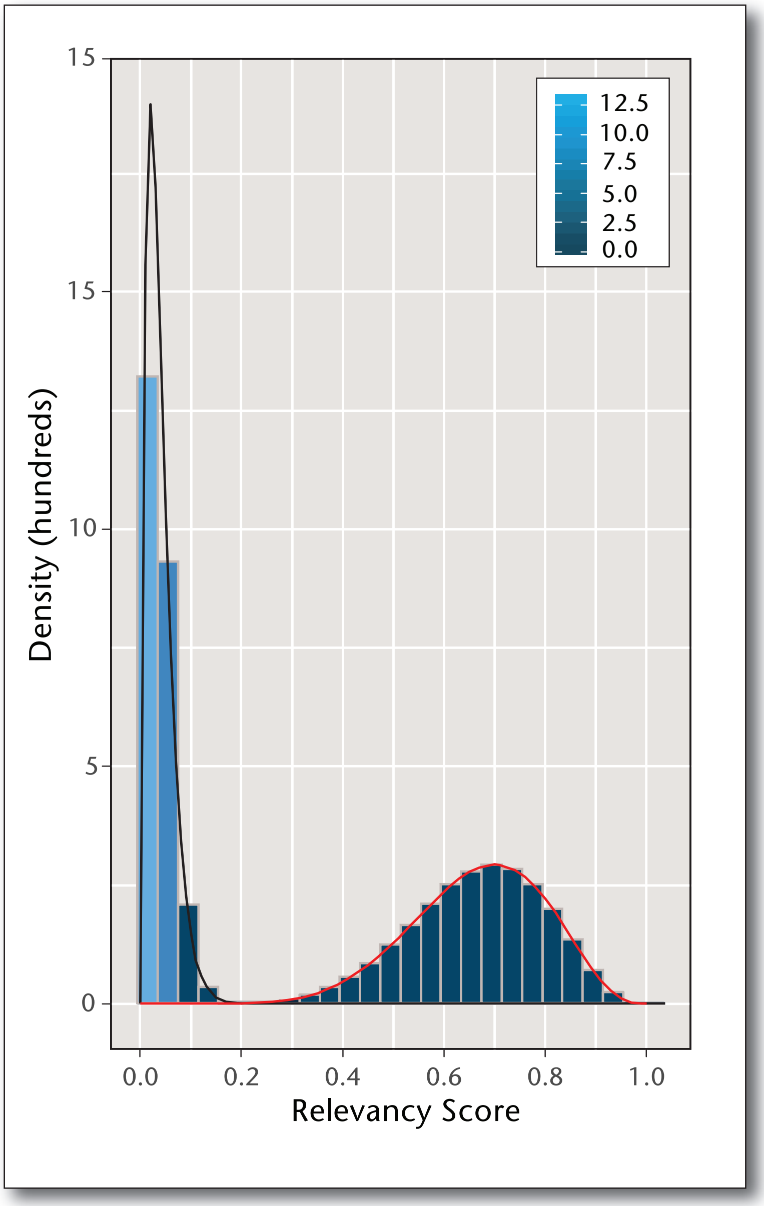

Figure 3. Scaling Relevancy Score by Beta Distribution.

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 skill 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

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

Table 5. Skill Tagging Results per Confidence Score Level.

The correlation between confidence score and approval rate is 0.81.

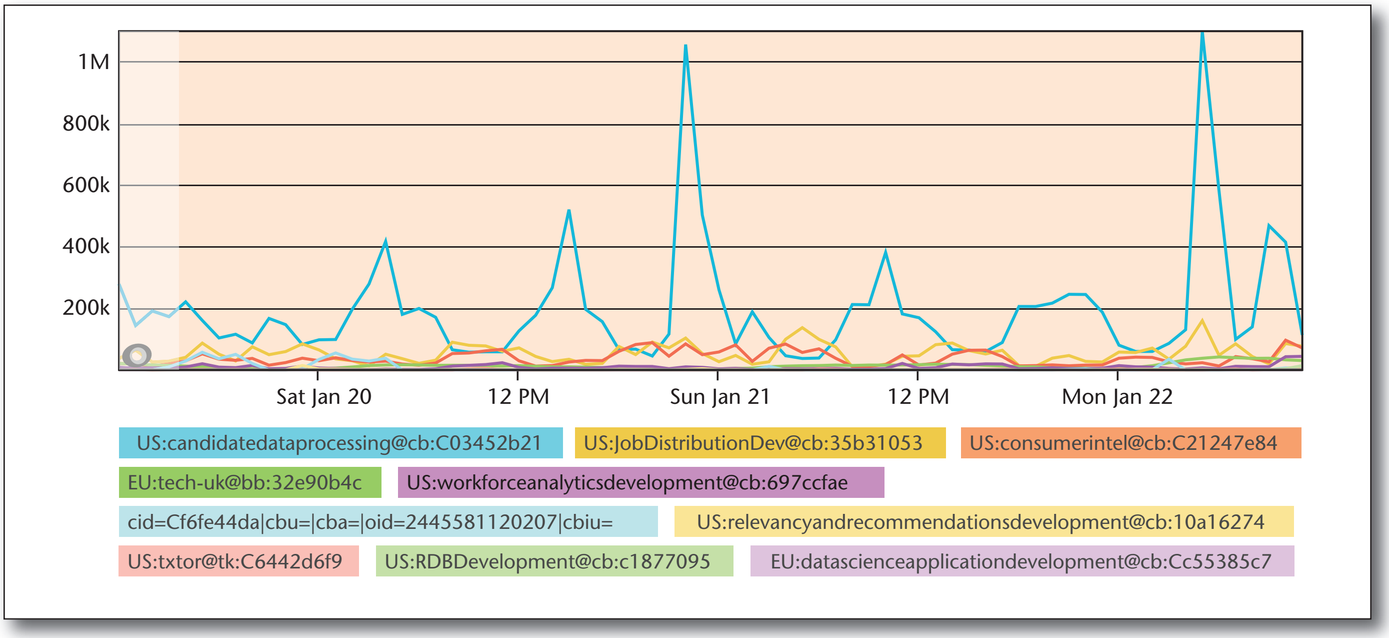

Figure 4. Skills Traffic Over a Three-Day Window, Broken Down by Caller.

The y-axis indicates the total calls per 30 minutes.

all been renamed for clarity. The codebase was upgraded to Java 8 and uses new features in the latest version of the Java SDK, such as optionals and the streams API, for more expressive and maintainable code.

The development team that deploys and operates the SKILL 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 CB and handles application startup, request and response deserialization, 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.

The SKILL service has been available for production use within CB's technology department for more than two years at the time of writing. In this time, a large number of development teams have found applications for the service.

Figure 4 shows a graph of production traffic to the service over a three-day window, broken down by caller. In total, the service provides skill tagging for over a dozen applications within the CB ecosystem. Traffic patterns vary per customer: some have higher volume, some send spiky bursts of traffic, and so on. Figure 5 shows a graph of production traffic for a single application, CB's demand data processing system, which runs in very large batches and creates massive amounts of traffic in short bursts. This caller's traffic was omitted from the graph in Figure 4 to ensure visual legibility. Even during our highest traffic periods, the SKILL service remains highly performant, with a 0.00 percent error rate and a 99th-percentile response time of 35ms. In the past year, we have been able to tune our performance and scalability to these levels by moving the service to a Docker-based containerized infrastructure, which allows us to bring up new instances in seconds during traffic spikes, and also reduces operational overhead costs.

Scaling up our server fleet to handle these traffic spikes smoothly proved quite difficult. Our first solution to this was to 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 SKILL service 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.

Usage and Maintenance

After maintaining the SKILL service in production for some time, we received customer feedback indicating a desire for a service that would return related skills for a skill. We were able to develop and deploy this functionality in a short amount of time and with

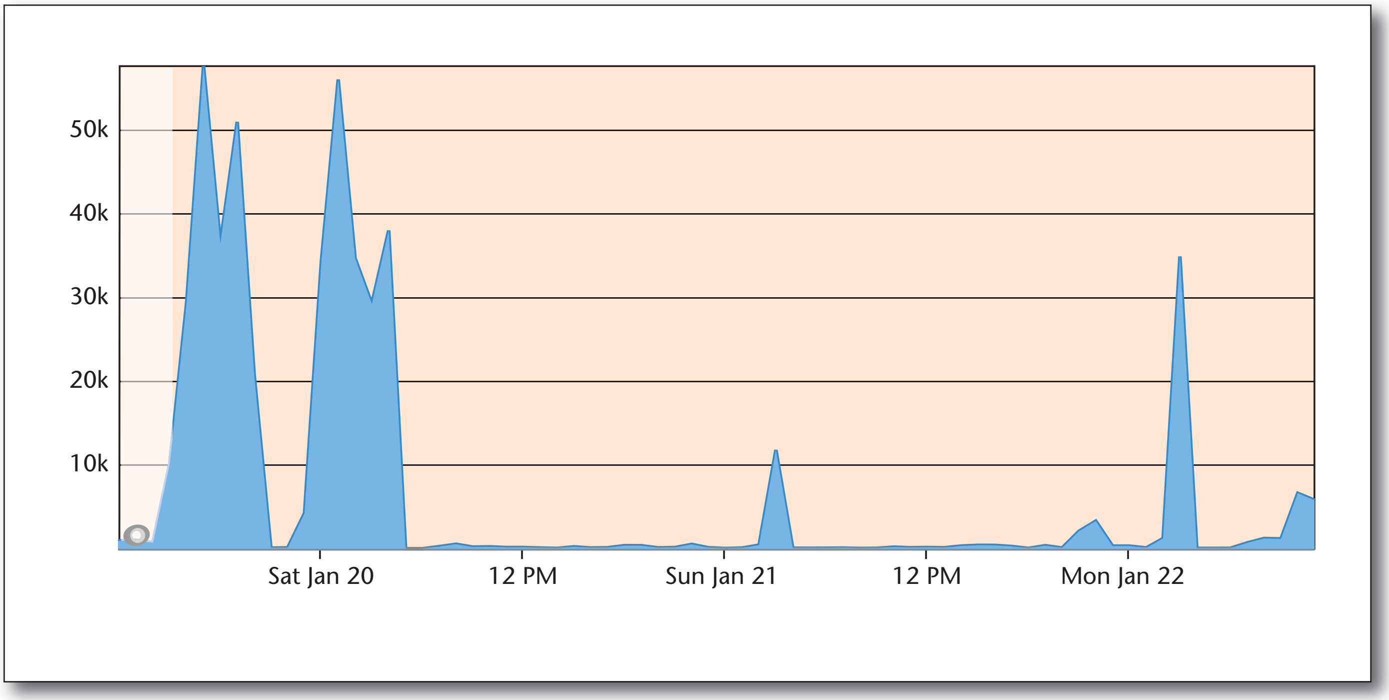

Figure 5. Skills Traffic Over a Three-Day Window.

Limited to one single caller who executes requests in massive batches. The y-axis indicates total calls per 30 minutes.

only a minor amount of development effort by reusing existing code from the skill extraction engine. In fact, mappings from normalized skills to their associated skill lists were already a feature of our data set. This new functionality was simply a matter of (1) normalizing the entered skill, and (2) returning an ordered list of its associated skills. Today, this feature adds customer value by enabling semantic expansions of candidate search queries written by recruiters, and by providing valuable data to hiring managers during the prehire process. Applications of this feature include offering career-path suggestions to job seekers and aiding job posters with fine-tuning description sections in job postings.

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 skill disambiguation mechanism. Continuous feedback loops from users also helped in removing noise from the taxonomy. Skill 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 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.

As part of operating and maintaining the SKILL service in production for a wide variety of internal customers, we have worked to develop a fast and reliable feedback loop to enable continuous iterative improvements on both the skill taxonomy and the extraction algorithms. To simplify maintenance processes for upstream customers, we treat each version as immutable and increment a version number whenever a change is made to either the taxonomy or the extraction logic. We support up to two versions at a time, and when a third is released, a retirement date is announced for the oldest live version. Over time, our modifications to the taxonomy naturally become less dramatic and more incremental: a new taxonomy version is now likely to be more than 99 percent identical to its predecessor. As a space optimization for our production servers, we developed a reusable library for persistent collections in Java — in other words, versioned collections to efficiently store multiple, mostly redundant versions of a data set without duplicating any data. Without this functionality, supporting two or three versions would essentially double or triple our memory requirements, significantly increasing operational costs.

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 socioeconomic 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 both from our internal stakeholders and from external customers on areas for improvement, and we are currently researching several future directions for the system. We plan to improve it by supporting case-sensitive tagging to minimize false positives and building a more comprehensive skill hierarchy. We will also support emerged, established, and saturated skill categorization. As we expand the service in various international markets, we will also support multilingual skill tagging and taxonomies. To support our compensation analytics and career path efforts, we plan to extend the service with skills proficiency, expertise inference, and skill effort capabilities.

Notes

1. www.bbc.com/news/education-25714313
2. money.cnn.com/2016/02/09/news/economy/america-5-6-million-record-job-openings/
3. www.wsj.com/articles/indias-skills-shortfall-challenges-modis-manufacturing-vision-1470653407
4. ec.europa.eu/esco
5. www.careerbuilder.com/
6. www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/ a11-smart-stop-list/english.stop
7. en.wikipedia.org/w/api.php
8. www.bls.gov/soc/2010/2010_major_groups.htm
9. developers.google.com/web-search/docs

References

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. New York: Association for Computing Machinery. doi.org/10.1145/2645710.2645729

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. Paper presented at the Graph-Based Methods for Natural Language Processing workshop, 18 October, Seattle, WA.

Luo, Q.; Zhao, M.; Javed, F.; and Jacob, F. 2015. Macau: Large-scale Skill Sense Disambiguation in the Online Recruitment Domain. In *Proceedings of the 2015 IEEE International Conference on Big Data,* 1324–1329. Piscataway, NJ: Institute for Electrical and Electronics Engineers. doi.org/10.1109/BigData.2015.7363890

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. Unpublished Manuscript. arXiv preprint arXiv:1301.3781. Ithaca, NY: Cornell University Library.

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. Stroudsberg, PA: 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. Stroudsberg, PA: 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 *Social Computing for Workforce 2.0: Papers from the 2013 ICWSM Workshop*, AAAI Technical Report WS-13-02, 20–23. Palo Alto, CA: AAAI Press.

Wang, Z.; Li, S.; Shi, H.; and Zhou, G. 2014. Skill Inference with Personal and Skill Connections. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING),* 520–529. Stroudsberg, PA: Association for Computational Linguistics.

Zhao, M.; Javed, F.; Jacob, F.; and McNair, M. 2015. Skill: A System for Skill Identification and Normalization. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence,* 4012–4018. Palo Alto, CA: AAAI Press.

Phuong Hoang is a data scientist in the Data Science R&D group at CareerBuilder. His research lies in the field of natural language processing and machine learning with applications to human capital management domain. He attended North Carolina State University, where he earned his BS in financial mathematics, and his M.S. and Ph.D. in applied mathematics, specializing in machine learning applications for medical diagnostics and sports analytics. He currently resides in Atlanta, Georgia, USA.

Thomas Mahoney is the manager of candidate and data services at CareerBuilder, where he oversees a group of teams focused on building out and maintaining classification, data enrichment, and candidate management web services to power CareerBuilder's products. He has worked in the human capital management space for the past five years and is passionate about building fast, reliable, and highly scalable microservices that deliver high customer value. He holds a B.S. degree in computer science from the Georgia Institute of Technology and currently resides in Roswell, Georgia, USA.

Faizan Javed is a manager of data science at CareerBuilder, where he leads the Data Science group responsible for data enrichment technologies such as knowledge bases, entity taxonomies and relationships, data standardization, and deduplication and normalization algorithms for the online recruitment domain. He has almost 10 years of industry experience in diverse domains with multiple technology stacks. Faizan has over 30 publications in areas ranging from data science and machine learning to software systems and model-driven engineering. His current area of focus is the application of data science to end-to-end human capital management processes. He holds an M.S. degree in computer science and bioinformatics, a Ph.D degree in computer and information sciences, and a certificate in technology entrepreneurship, all from the University of Alabama in Birmingham, Alabama, USA.

Matt McNair, is vice president of global services strategy, where he focuses on the edges of Careerbuilder's products, ensuring that they drive recruiter efficiency by integrating well with each other and with the tools that recruiters use daily. His 12 years of experience in the recruitment space has made him passionate about applying data science, running high-scale microservices, and bringing it all together into a recruiter-friendly sourcing product. He currently resides in Atlanta, Georgia, USA.