LinkedIn Skills: Large-Scale Topic Extraction and Inference

Mathieu Bastian
LinkedIn Corporation
2029 Stierlin Ct.
Mountain View, CA 94043
mbastian@linkedin.com

Matthew Hayes
LinkedIn Corporation
2029 Stierlin Ct.
Mountain View, CA 94043
mhayes@linkedin.com

William Vaughan
LinkedIn Corporation
2029 Stierlin Ct.
Mountain View, CA 94043
wvaughan@linkedin.com

Sam Shah
LinkedIn Corporation
2029 Stierlin Ct.
Mountain View, CA 94043
samshah@linkedin.com

Peter Skomoroch
LinkedIn Corporation
2029 Stierlin Ct.
Mountain View, CA 94043
pskomoro@linkedin.com

Hyungjin Kim
LinkedIn Corporation
2029 Stierlin Ct.
Mountain View, CA 94043
ekim@linkedin.com

ABSTRACT

"Skills and Expertise" is a data-driven feature on LinkedIn, the world's largest professional online social network, which allows members to tag themselves with topics representing their areas of expertise. In this work, we present our experiences developing this large-scale topic extraction pipeline, which includes constructing a folksonomy of skills and expertise and implementing an inference and recommender system for skills. We also discuss a consequent set of applications, such as Endorsements, which allows members to tag themselves with topics representing their areas of expertise and for their connections to provide social proof, via an "endorse" action, of that member's competence in that topic.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

inferring attributes; social networks; recommender system; folksonomy

1. INTRODUCTION

LinkedIn, the world's largest professional online social network, consists of a collection of profiles that represent a member's professional identity. A profile contains a member's current and past work history, education, and projects, among other pieces of his identity. For many members an important part of that identity is the set of skills and talents for which they are known. The "Skills and Expertise" section is a part of the profile that allows the member to display this information, as shown in Figure 1. We refer to this section as "Skills and Expertise" rather than just "Skills" because it can also include domains of expertise, which are not strictly skills. For instance, "Natural Gas" is a domain of expertise an energy professional might want to be known for. The member may add skills to his/her profile either by manually entering them, in which case a type-ahead prompt is provided, or by accepting recommendations of inferred skills.

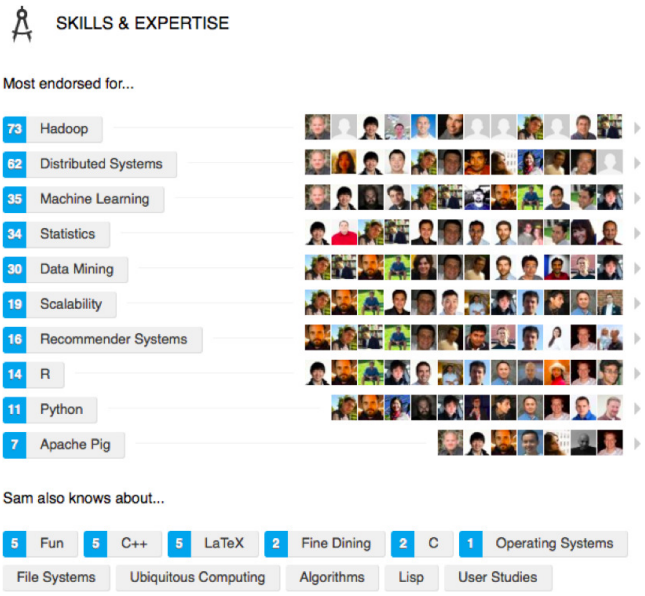

Figure 1: Example skills and expertise section with endorsements

There is an important value in helping members enter their skills and expertise in the system in a standardized fashion and the goal of the skills feature on LinkedIn is to facilitate that. Standardized entities help members be better found by increasing the coverage in search engines. It also helps job matching algorithms, as the definition of a member's competencies is crisper and less noisy.

Prior to the introduction of the skills and expertise section on profiles, we chose to create a folksonomy of skills that rep-

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.

ACM 978-1-4503-2668-1/14/10 ...$15.00.
http://dx.doi.org/10.1145/2645710.2645729.

resented the full breadth and diversity of members' skills on LinkedIn so an exhaustive type-ahead could be provided. We evaluated several publicly available skill listings, but none were sufficiently comprehensive or detailed. LinkedIn contains hundreds of millions of member profiles representing professional interests as diverse as ballet, machine learning, air traffic control, and scuba diving. Manually creating this taxonomy was an infeasible task due to the size and diversity of these profiles.

We decided that the best way to create this taxonomy was to derive it from the data of member's profiles directly, thus opting for a folksonomy. Creating a data-driven solution was feasible, but still challenging. The hundreds of millions of profiles contained millions of keywords that might or might not be skills. Additionally, many of these potential skills are duplicates. For example, "Java programming" and "Java development" are both equivalent to the skill "Java". Also, many of these phrases have different meanings depending on context. The phrase "Organ" on a musician's profile probably means a musical instrument while the same phrase on a physician's profile likely refers to a part of the body.

To create this folksonomy from the data we broke the problem down into three phases: discovery, identification of the potential candidate skill phrases; disambiguation, where we attached the appropriate context to each meaning of each phrase; and deduplication, where we merged phrases that were semantically identical into a single skill. Our implementation of this approach uses a mixture of established and novel approaches in machine learning techniques and crowdsourcing.

Once the folksonomy was in place and the initial skills and expertise section had launched, we recognized a need to provide recommendations to allow members to more easily enter their skills and expertise. The intuition was that it would be more effective to ask members to confirm that they have a set of skills rather than asking which skill they have and provide a blank form. We implemented an inference system that calculates the likelihood of each member having each skill and presented these inferences as recommendations. The lack of initial skills data created a cold-start problem but we designed a solution leveraging the rich set of other profile attributes. The introduction of this recommender system significantly increased the number of members adding skills.

The main contribution of this work is a description of a large-scale folksonomy generation of skills and expertise and the system to infer and recommend them to members. We describe the challenges around discovery, disambiguation, and deduplication to create the folksonomy and discuss our approach to build a collaborative filtering (CF) skills recommender system based on profile attribute data. We also discuss implementations of a diverse set of applications of this folksonomy and inference algorithm.

2. SKILLS AND EXPERTISE

The purpose of the skills feature on LinkedIn is to provide a definitive section on a member's profile to list that member's skills and expertise. Prior to the creation of this section, information about a member's skills was spread among a number of sections and embedded in free text. To assist the member in standardizing this information in the skills section we first created a folksonomy of skills to provide a type-ahead. After observing a lower than desired interaction with the type-ahead, we then created a recommender system to allow members to more easily add skills and expertise to their profile.

2.1 Folksonomy creation

Our goal was to create a large enough list of skills and expertise that members might choose from to add on their profile. This list should contain skills and expertise that everyone might expect in a professional context such as "Management" or "Public Speaking" but also rare competencies such as "Atmospheric Physics" or "Economics Education". To accomplish this goal, we first considered using an existing taxonomy. We evaluated a number of available lists, but found none that were sufficiently broad in scope. Instead of attempting to find a sufficient number of experts to construct such a list, we decided to discover the list from the data available in the members' profiles and created a folksonomy [9].

Folksonomies are often used within Internet communities to categorize content [12] and are directly edited through users' contributions. In our case, we already had skill information across all sections of a profile but it wasn't structured or organized so a list can be extracted. Identifying domain-specific terms from free text in order to create a folksonomy has been studied in [3] [18] [11]. In our case, we observed that in a particular profile section, the specialties section, a large number of members provided free text matching a simple pattern, a comma-separated list, and that the entries in this list were very frequently skills. The specialty section was a common section on LinkedIn profiles at the time of the initial extraction process, appearing on over 12 million profiles. We decided to use this observation to extract a list of candidate topic *phrases*. In this section, we prefer to use "phrase" instead of "keyword" as the entities we're looking for can have multiple words. The number of words was limited to six for display purposes.

2.1.1 Discovery: Entity extraction

The first step in identifying entities that would become skills was to extract phrases from profile specialty sections containing comma-separated lists of topics. The profile section was a free text field, which could contain complete sentences, lists of topics delimited by commas, bullet points, or other characters. We first aimed to recognize that the section was actually a comma-separated list and used a simple technique based on the punctuation frequency. We define Cf as the comma frequency in the text section and filtered out texts based on CT, a minimum threshold.

$$Cf = \frac{number\ of\ commas}{number\ of\ characters} > CT$$

Heuristically, we found that CT = 0.02 is a good trade-off to obtain a relatively consistent dataset for the case of specialties on LinkedIn profiles. By looking at the results, we found that a lower threshold allows too many sections which enumerate lengthy accomplishments rather than skill keywords. We also found that a higher threshold penalizes sections with multiple compound words such as "Supply Chain Optimization".

All sections with a sufficient Cf frequency were then tokenized to obtain phrases. A stop-word list was finally applied to remove phrases that are common parts of speech. This stop-list was augmented with a domain specific stop

list which removed commonly observed entity prefixes and suffixes. The remaining phrases of up to six words were preserved.

This phrase list contained the candidate skills along with other entities. The next step was to remove phrases, which represented other known types of entities, such as companies, degrees, and titles. Fortunately, these entities could be extracted from member profiles as well, but in a structured format. However, simply filtering out every phrase that matched with a known entity would have been too aggressive because some skills overlapped with other known entities. For instance, "Oracle" is both a domain of expertise and a company name and "Computer Science" could be both a degree and a skill. To address this issue we compared the phrase frequency of occurrences in the specialty section with the frequency of occurrences in other sections of the profile to create a ratio. We then introduced a threshold for which entities were placed in the stop list if their frequency in the specialties section wasn't sufficient. The threshold was chosen heuristically by comparing the results at different values.

Finally, we counted the number of profiles in which each phrase occurred and removed all candidates that were below a minimum frequency threshold of twenty. This threshold was chosen to be low enough to allow rare skills to go through but high enough to remove much of the long tail noise. This filtering produced a rich dataset of about 150,000 skill phrases. As expected, the number of occurrences is long-tail distributed with a few thousand common phrases and a majority of rare ones. In fact, only 6% of skill phrases represented approximately 80% of occurrences.

2.1.2 Disambiguation: Clustering to provide context

After finding an initial set of skill phrases, we had to disambiguate those phrases that had multiple meanings dependent on their context. For example, the phrase "organ" has a separate meaning in the contexts of music and anatomy and is a valid expertise in both cases. The first step of the disambiguation process was to generate a set of related phrases for each candidate skill phrase. Related phrases are those skill phrases that appear often with another skill phrase on profiles. The intuition is that an ambiguous phrase would have distinctly different sets of related phrases in each context. In other words, a relationship between two phrases was induced by the neighborhood of these phrases. For instance with "organ", the term "piano" appears frequently in the musical meaning and rarely co-occurs in the medical meaning, where phrases such as "surgery" are common. An example of such a rare co-occurrence might be a physician's profile where piano is also listed as a skill. This technique is more formally described in the query expansion chapter of [2]. Finally, clustering was applied to distinguish the distinct senses for each phrase.

More formally, let p_i be a skill phrase, d_k a member profile, $P = \{p_1, p_2, ..., p_t\}$ be the set of all skill phrases and $D = \{d_1, d_2, ..., d_{t'}\}$ be the set of all member profiles. Let $f_{i,k}$ be the occurrence of the phrase p_i in a section of a member profile d_k, such as the previously mentioned specialties section. The occurrence count F_i of phrase p_i in D is defined as

$$F_i = \sum_{k=1}^{t' \in D} f_{i,k}$$

In addition, let $f_{i,j,k}$ be the co-occurrence of the phrases p_i and p_j on the a same section of a profile d_k. The co-occurrence count between two phrases i, j is then defined as

$$F_{i,j} = \sum_{k=1}^{t' \in D} f_{i,j,k}$$

The Jaccard similarity $S_{i,j}$ between two phrases is then calculated as

$$S_{i,j} = \frac{F_{i,j}}{F_i + F_j - F_{i,j}}$$

For each phrase p_i, the top-n most related phrases are kept based on the highest Jaccard similarity with n being set in advance and we define $R_{i,n}$ as the vector of size n of related phrases $r_{i,j}$ for p_i with $j \in P$. Other standard similarity metrics were tried but Jaccard was heuristically chosen as it seemed to yield the best results and remained simple to implement. R_i is then transformed into a matrix M_i by using R_j as a column for each element $r_{i,j}$ of the vector. The $S_{i,j}$ similarity is inserted for each $r_{i,j}$ cell of the resulting M_i matrix.

$$M_i = \begin{bmatrix} S_{1,1} & S_{1,2} & \dots & S_{1,n} \\ S_{2,1} & S_{2,2} & \dots & S_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ S_{n,1} & S_{n,2} & \dots & S_{n,n} \end{bmatrix}$$

This matrix M_i is next transformed by calculating the cosine-similarity $T_{i,j}$ [15] for each cell using the R_i and R_j vectors. Each cell of the resulting matrix N_i is now the cosine-similarity score between the related phrases of p_i and p_j.

$$N_i = \begin{bmatrix} T_{1,1} & T_{1,2} & \dots & T_{1,n} \\ T_{2,1} & T_{2,2} & \dots & T_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ T_{n,1} & T_{n,2} & \dots & T_{n,n} \end{bmatrix}$$

Clustering was then applied on the N_i matrix to distinguish multiple senses associated with the $p_i \in P$ phrase. We used the SVD technique to find the top components for each phrase by dimensionality reduction. Heuristically, we found that the first six or seven components yielded the best results. Finally, we ran a KMeans algorithm to identify clusters of phrases. The best K was chosen dynamically based on the inter-cluster distance and a penalty to avoid having too many clusters. We observed that most phrases would only have one dominant cluster but ambiguous terms would often have two or three groups.

To aid in interpretability of the clusters, an industry label was attached to each of them. Each member profile possesses an industry attribute entered by the member at registration. To determine the cluster industry label we simply took the most frequent industry among the member profiles from which the phrase was found. For instance, the phrase "organ" appears in clusters in the industries "music" and "medicine".

2.1.3 Deduplication: crowdsourcing

In the previous sections we described the generation of candidate skill phrases and the disambiguation of these phrases to multiple senses. But many of these phrases were semantic duplicates of each other. This duplication results from

members entering synonyms in phrases such as "java development" and "java programming", making typos such as "government liaison" and "goverment liason", abbreviating words or referring to acronyms, such as "B2B" and "Business to Business", and a number of other possible scenarios.

Duplicate skills hinder the user experience. For example, a user may be unsure whether "Java programming", "Java development", or simply "Java" is the preferred way to list a skill. However, manual identification of all duplicates across the large number of phrases would be extremely difficult. To address this problem we investigated a crowdsourcing solution to handle the deduplication.

During a small-scale experiment, we found that more than two thirds of the skill phrases we considered could be matched to a Wikipedia page describing the topic using a simple Web search. We also found that in some instances many phrases were related to the same page and that, in those cases, those phrases referred to the same skill. Based on these findings, we decided to use relationship to the same Wikipedia page as a criterion for deduplication.

In order to do this mapping at scale, we built a crowdsourcing task using Amazon's Mechanical Turk [1] platform. Each task asked the worker to identify the best Wikipedia page for a given phrase. Additional information like the industry cluster and the related phrases were provided to give the worker sufficient context to make an informed decision. Figure 2 shows an example for the "organ" phrase.

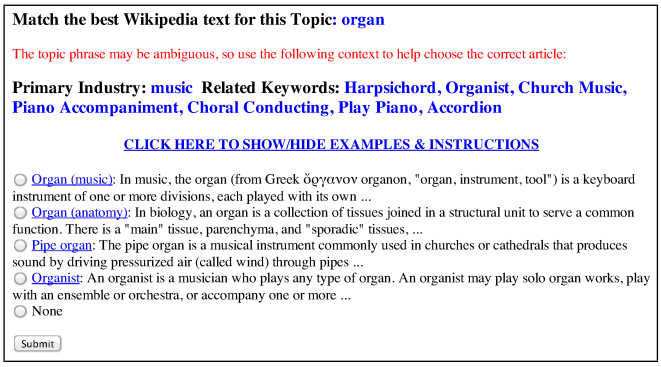

Figure 2: Crowdsourcing task for the "organ" phrase

Once this task was complete, phrases which matched to the same Wikipedia page were combined into a single *skill*. This skill was given the label of its most frequent phrase. This process resulted in a mapping of the 150,000 skill phrases to approximately 50,000 skill topics. Finally, this list of skills was curated to remove obvious errors and then served as a type-ahead to members editing their profile.

2.2 Skills Inference and Recommendation

In the previous sections, we described how we built the folksonomy of standardized skills. This list was used by members to edit their profile and was an improvement over no standardization at all but didn't meet the desired profile completeness growth objective. Therefore, we decided to create a skill inference algorithm and directly suggest additional skills to members through a recommender system [13]. A system of this kind should be personalized and aims to only suggest the skills members have. When presented with skills recommendations, we assume members would only add the skills they possess and reject the ones that were unknown or not relevant. This problem is often encountered in discovering latent attributes in social network profiles [7] [10] [1]. These techniques use a variety of explicit and implicit profile attributes and often leverage the homophily characteristic of social networks.

[1]https://www.mturk.com

Type	Coverage	Example
Title (Headline)	89%	Product Manager
Function	70%	Engineering
Industry	100%	Healthcare
Title (Employment Position)	53%	Product Manager
Company	46%	LinkedIn
Group membership	29%	Healthcare Professionals
Field of Study	24%	Computer Science

Table 1: Profile attributes dataset with coverage

Unlike common recommender systems, which recommend a large set of items such as movies or products, the number of skills relevant to a member is relatively limited. In fact, the maximum number of skills a member can add to his LinkedIn profile is fifty. In this regard, this problem is quite similar to recommending tags for content such as photos or articles, which is discussed in [4] [16] [8]. In our case, the content would be the member's profile and the tag the skill. These systems often try to find the best trade-off between popularity and personalization, and the same is true in our case. On one hand, the most popular skills are a small fraction of the approximately 50,000 distinct skills in the folksonomy, but a large proportion of members list them. On the other hand, rare and more specific skills might be more descriptive and appealing to members as they more accurately describe their competencies.

We investigated different approaches to solve this problem and ultimately developed a collaborative filtering (CB) solution. There was limited information available about the skills themselves so content-based are difficult to apply directly. However, a member's profile normally includes explicit data such as employment history, education information or group affiliations and we noticed that members who share these profile attributes were likely to have similar skills and expertise. For instance, if a member is an *A*pple employee, it is likely that he/she has the skill "Mac OS" because other employees at Apple have this skill. Similarly, most members of an interest group such as "Healthcare Professionals" should have many skills in common. Therefore, we started building a training dataset by collecting standardized profile attributes that are likely to be shared by members with similar skills along with the skills these profile list.

At the time of this extraction, the number of members with skills was too limited and we experimented adding the skills extracted from the free text specialty section to complete the training set. Although this introduced some noise it helped create a larger training set to start with. A description of this dataset is in Table 1 with an indicative coverage metric, which represents the number of members who have a particular attribute in the dataset.

We first considered using a Logistic Regression model to estimate the likelihood that a member has a particular skill. Since training a model per skill was impractical given the size of the folksonomy and unsuitable for ranking applications, we looked into an alternative approach - creating a unique regression model based on collaborative filtering. This approach would require transforming the profile attributes and creating a user-similarity based metric in order to associate skills with members. Given the diversity of the attributes we planned to use, this approach seemed to create a quite difficult undertaking. Instead, we looked into using a simpler naive Bayes classifier model which is suitable for both large vocabularies and large number of classes. Although our profile features are not truly independent, naive Bayes often compares with more sophisticated classifiers [14] in terms of performance.

2.2.1 *Naive Bayes Classifier*

We identified standardized profile attributes such as the company, title, industry, field of study or group membership to be useful in inferring skills or areas of expertise. Given a profile that contains a set of these uniquely identified attributes $a_1...a_n$, we define the likelihood to have a skill as $P(s|a_1...a_n)$ where $s \in S$ and $S = \{s_1, s_2, .., s_n\}$ the folksonomy of skills. Let $n(a_i, s)$ be the number of co-occurence of the attribute a_i and the skill s on member profiles in our training dataset and $n(s)$ the total number of member profile with that skill. Using the naive Bayes formula, we define

$$P(s|a_1, \ldots, a_n) = \frac{1}{Z} P(s) \prod_{i=1}^{n} P(a_i|s)$$

where $Z = P(a_1, \ldots, a_n)$, $P(s)$ is the prior probability of a skill $s \in S$ and $P(a_i|s)$ is the probability of a_i given s. We define

$$P(a_i|s) = \frac{n(a_i, s)}{n(s)}$$

When calculating the most likely skill a member has, it gives

$$\underset{s \in S}{\text{argmax}} \; P(s) \prod_{i=1}^{n} P(a_i|s)$$

Because $n(a_i, s)$ might be missing for some feature and skill, we're using Laplacian smoothing to avoid zero predictions. We then have

$$P(a_i|s) = \frac{1 + n(a_i, s)}{|W_i| + n(s)}$$

where $|W_i|$ is the vocabulary size for attribute i. Our implementation used log probabilities to avoid underflow.

Using a naive Bayes classifier to infer skills required strong prior associations between members' skills and the profile attributes used in the training phase. While the set of members with each attribute type varies, most members' profiles contain at least a couple of distinct attributes. For instance, most of them have an industry, which has about 150 distinct categories, as it's asked at registration time. Therefore, this approach allowed producing an inferred skills ranking for almost every member on LinkedIn. That said, the ranking quality varied from simply the most frequent skills in your industry to a more accurate prediction as the number of attributes used as feature increased.

2.2.2 *Feature selection*

Using this naive Bayes model, we found that the number of features could easily become unmanageable. Because profile attributes include entities like interest groups or companies, the number of distinct attributes can grow very large. For instance, there are more than 3 millions distinct companies on LinkedIn for which members can have worked. Not only does the expansion of these attributes become unwieldy to work with, but the large number of very rare or very common occurrences might also decrease the prediction accuracy, as presented in [5]. Therefore, we aimed to reduce the number of features using a feature selection technique. We compared two feature selection techniques with the objective of finding the best trade-off between accuracy and the size of the training data.

At first, we used a frequency filter to remove attribute and skill occurrences below a certain threshold but found that this technique overly penalize rare skills when set too aggressively.

We then opted to use a mutual information calculation, which is often used with naive Bayes to evaluate which feature to select. The mutual information between a skill s and an attribute a_i class measures how much the presence/absence of an attribute contributes to a correct classification of a skill. Formally:

$$\sum_{e_s \in \{1,0\}} \sum_{e_a \in \{1,0\}} P(U = e_s, C = e_a) log_2 \frac{P(U = e_s, C = e_a)}{P(U = e_s)P(C = e_a)}$$

where U is a random variable that takes values $e_s = 1$ (profile has skill) and $e_s = 0$ (profile doesn't have skill) and C is a random variable that takes values $e_a = 1$ (profile does have attribute) and $e_a = 0$ (profile doesn't have attribute). We calculated the mutual information between each skill s and attribute a_i and defined this result as our utility function to filter out (feature,skill) pairs with low information gain. We then used the utility score distribution to select a threshold. We tested multiple thresholds and compared the recall@50 on a sample of profiles and features for a threshold set at the 1st, 5th, 10th, 25th and 50th percentile. Figure 3 is showing the result of this comparison.

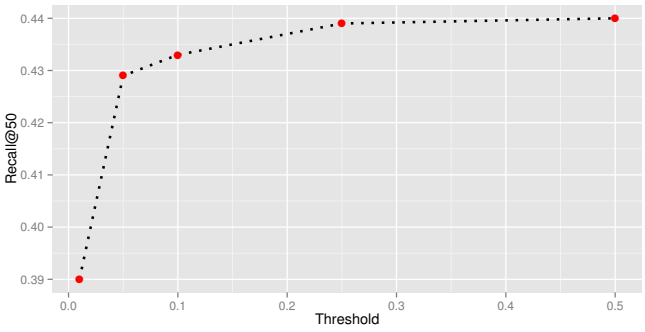

Figure 3: Mutual information threshold effect on classification performance

We observed that the classification performance is holding out well even with only a fraction of the complete training data. That said, the recall is calculated using members who already have skills so there was a bias in this dataset. We finally opted for a threshold at the 25th percentile as it main-

tained performance, did not penalize rare entities too much and reduced the training data to a manageable size.

2.2.3 Evaluation

We first evaluated the accuracy of the naive Bayes classifier by comparing predicted skills with actual occurrences using a holdout dataset. The classification result is a sorted list of inferred skills and this order is being used by the recommender system. We primarily cared about the quality of this ranking and designed our evaluation methodology around that. True positives were inferred skills members had in the skills section of their profile and our proxy for true negatives were inferred skills members do not have. Whether the skill phrase was present in the profile free text was not taken into account in the evaluation. We could use information retrieval metrics such as MAP [2] to directly evaluate ranking performance but also precision and recall at a particular k position. The area under the ROC curve is also a valid evaluation but can only be performed per skill, as scores are comparable. Although the classifier output is not a probability we could use a normalized score per skill to obtain an ROC curve. Figure 4 shows the ROC curve for the "Hadoop" skill in sample.

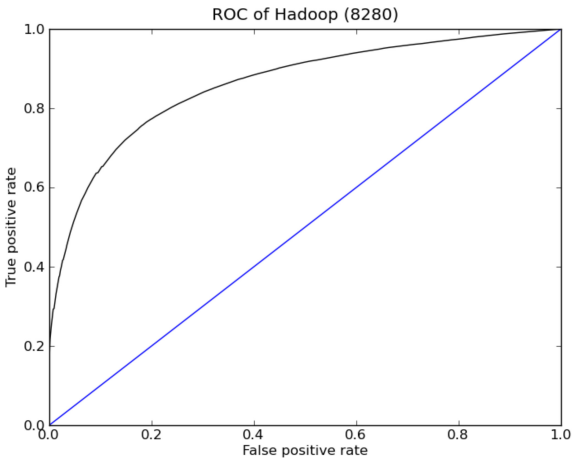

Figure 4: ROC curve for the "Hadoop" skill

The average AUC on ROC curves on all skills with a minimum occurrences of 100 members was 0.77. Figure 5 shows the distribution of these scores.

We observed that more specific skills obtain better AUC results. For instance, technical skills, such as "Hadoop", are well above the average but soft and vague skills such as "Ability to learn" obtain AUC scores around 0.5. This observation can be explained by studying how skills and attributes co-occur. Soft skills tend to co-occur with a much broader set of attributes whereas technical skills are often only co-occurring with a small set of companies, titles or groups. We also looked at the impact of different feature set on the classification performance. Table 2 shows the classifier recall for different combination of features as an illustrative example.

2.2.4 Results

We applied the skills inference algorithm to suggest to members skills to add to their profile and evaluated. This

[2]Mean Average Precision

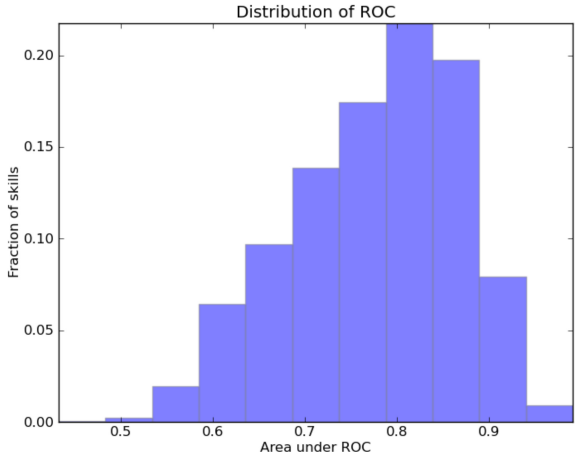

Figure 5: Distribution of ROC AUC across all skills

Feature Set	Recall@50
industry	0.45
industry+company+function	0.60
industry+company+function+title	0.66
industry+company+function+title+group	0.71

Table 2: Classification performance with different feature sets

suggested skills task was presented to a large set of members on LinkedIn in a system that invited them to complete their profile. Because a large majority of members have at least one standardized profile attribute, generally an industry - we could provide skills suggestions for >95% of LinkedIn members, even if their profile was not in English. We primarily looked at two metrics to evaluate the effectiveness of the system: conversion and acceptance rate. The conversion is defined by the ratio of members adding at least one skill over the members who saw the task. The acceptance rate is defined as the ratio of skills added over skills impressed.

One of the key results was that the recommender system led to far greater conversion of members adding skills to their profiles. Specifically, we compared two different user interfaces as shown on Figure 6. The user interface with suggestions led to a 49% conversion compared to 4% with a simple blank field with type-ahead.

We also performed experiments regarding the number of skills to suggest. We started with three skills and found that there wasn't a significant penalty in adding more skills to suggest up to ten. We monitored the conversion and acceptance rate but also the dismissal rate, which represented the percentage of suggestions members dismissed. We were concerned that too many suggestions would overload the member with information and halt the healthy "pick-n-choose" behavior we were observing. Indeed, we wanted members to continue to dismiss the suggestions they did not found relevant. Figure 7 compares the frequency of skills added during an A/B test, which shows either five or ten suggestions.

We observed that 57% of members added five skills when five were shown and 45% of members added ten skills when ten were shown. Therefore, we did not observe an increase in the frequency of members adding all the presented suggestions.

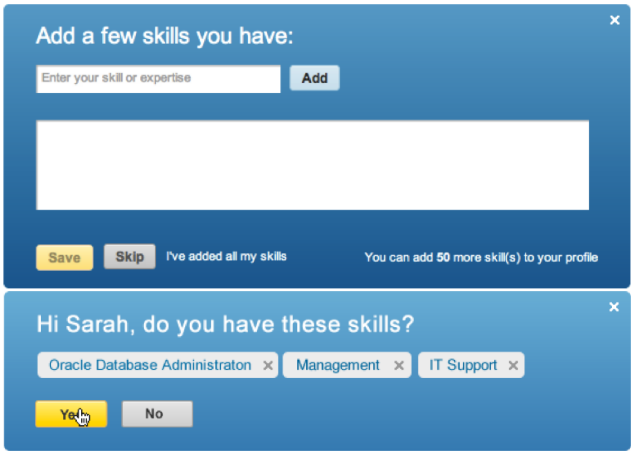

Figure 6: Add skills with just typeahead: 4% conversion. Add skills with recommendations: 49% conversion

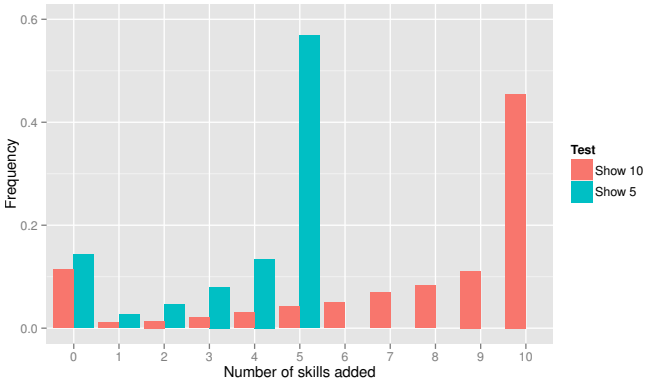

Figure 7: Skills added with 5 or 10 suggestions shown

3. IMPLEMENTATION

The skills data pipeline is running on Hadoop, an open source implementation of MapReduce, as part of LinkedIn's big data ecosystem [17]. MapReduce provides a framework for processing big datasets on a large number of commodity computers and further simplifies the process of writing parallel programs by providing the underlying infrastructure, failure handling, and simple interfaces for programmers.

Incoming query logs and other activity-based tracking data is aggregated from production services using Kafka [6], a publish-subscribe system for event collection and dissemination. This log data can be easily represented as a set of tuples, which is a natural fit for Hadoop's distributed computing paradigm.

The skills pipeline consists of a series of automated MapReduce jobs, separated in two workflows: one for the folksonony creation and one for the inference. In total, there are over 150 MapReduce jobs, from initially pre-processing the profiles to performing entity extraction, disambiguation, deduplication and inference.

In addition, an online classifier is running as part of our inference service, which is classifying members with skills upon each profile update. This data is available via an API or as a stream for data consumers.

4. APPLICATIONS

We present a set of key applications that directly leverages the skills folksonomy and the inference system.

4.1 Endorsements

In September 2012, LinkedIn released the Endorsements feature, which allows members to provide social proof of their connections' skill or expertise in a given topic via means of a new social gesture, an "one-click endorse." The endorsing member's endorsement is then displayed next to the respective skill of the endorsed member's skill section. An example skills and expertise section with endorsements is shown in Figure 1.

One of the goals of the Endorsements product was to encourage those people who had few skills on their profile to add more. When a member A endorses another member B for a skill that member B does not yet have, member B is notified of that endorsement and prompted to accept it and add that skill to his/her profile. Towards this goal, a recommender system was built to suggest (*connection*, *skill*) pairs which a member might want to endorse.

An important aspect of this recommender system is the fact that it can create endorsement recommendations for skills that the endorsement's target does not yet have. In this process the results of the inferred skill algorithm are a primary feature in the choice of skills to recommend. The ability for members to suggest skills to other members via an endorsement added a social component into the skills recommendation and has proven a very effective technique to increase the number of members adding skills.

4.2 Query Expansion

For search applications, query expansion can be an effective technique to improve the search quality, especially in situations were the number of results are low. The idea is to expand the query to include additional terms as alternatives that are related to those within the query. A typical example of this technique is including synonyms of search terms. For example, a query consisting of just "car" could be expanded to include "automobile" as well. As these two terms are closely related, it is reasonable to return results matching either of them, perhaps with the one from the original query weighted more heavily.

One common type of query a member may perform on LinkedIn is searching by skill. For example, a recruiter may be seeking members with particular skills that they would like to reach out to. Query expansion can make their skill-based searches more effective by automatically including other closely related skills.

The key ingredient in making a skills-based query expansion work is a measure of how closely related skills are to each other. Fortunately the collection of member profiles and job postings, with the skills identified in them, provide this. If we assume that closely related skills are more likely to co-occur on the same document (i.e. profile, job posting), then we can use the frequency of co-occurrence to derive similarity scores between all the skills in the folksonomy. These scores can then be used to expand the skills-based queries to include skills that fall above a certain similarity threshold.

4.3 Job Recommendations

Effectively recommending jobs to members is important for a professional networking site. An effective feature in

such a recommender system could be the skill similarity between a job and a member. A member is likely to show interest in a job that has a skill profile similar to their own skill set. Some of the techniques from query expansion can be applied here.

We can begin by identifying skills within the text of member profiles and job recommendations. This produces a set of skills for each profile and job posting. This skill set can then be expanded with similar skills using the same query expansion technique described earlier. For each skill we can generate a score that indicates the relevance of that skill for the particular profile or job posting. Several factors may influence relevance. Skills that appear explicitly are more relevant than skills that are inferred due to query expansion. Skills that are closely related and co-occur in the same profile or job posting have a reinforcing effect that boosts relevancy of those skills. From the set of skills and their associated relevancy scores we can construct skill vectors that summarize how closely related each profile or job posting is to each known skill. Given these skill vectors we can compute similarity between the member and job posting using any number of metrics. This score can then be yet another important signal in a job recommender system.

5. CONCLUSION AND FUTURE WORK

In this paper we have presented the "Skills and Expertise" data driven feature that is part of LinkedIn. We have discussed the initial creation of the folksonomy, covering entity discovery, disambiguation, and deduplication. The combination of techniques presented has created a set of skills and expertise that covers a broad scope of domains with minimal need for editorial influence.

In addition, we have also presented the design of an inference and recommender system to suggest skills and expertise to members. This system allowed to rapidly expand the set of members with a qualified set of standardized skills on their profile. Subsequent applications have been able to leverage the work done in the extraction and inference of skills to provide additional value in multiple domains.

The folksonomy of members' skills and expertise is inherently dynamic. Continuing work is required to better understand how folksonomies such as this one evolve over time and how the knowledge encoded in them may be best utilized. Moreover, the expansion of this approach to non-English languages might be an interesting challenge. Better leveraging what we know about the skills from the Wikipedia link might also be a promising direction, especially regarding the disambiguation of the multiple meanings of a skill entity.

The inference system could also be expanded with more features and a more sophisticated approach. This approach was a good start with the data we had but with the recommender system being used we can collect real-world feedback data, which can be leveraged in a machine-learnt model. We also suspect that the skills members might want to add to their profile are slightly different from the skills we infer they might have - leading to a slightly different objective function. Additional features, which are not profile attributes could also be explored such as the frequency of the skill within the member's connections or bag-of-words.

6. ADDITIONAL AUTHORS

Additional authors: Sal Uryasev (LinkedIn Corporation, email: `suryasev@linkedin.com`) and Christopher Lloyd (LinkedIn Corporation, email: `clloyd@linkedin.com`).

7. REFERENCES

[1] F. Abel, Q. Gao, G.-J. Houben, and K. Tao. Analyzing user modeling on twitter for personalized news recommendations. In *User Modeling, Adaption and Personalization*, pages 1–12. Springer, 2011.

[2] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[3] F. Echarte, J. J. Astrain, A. Córdoba, and J. E. Villadangos. Ontology of folksonomy: A new modelling method. *SAAKM*, 289:36, 2007.

[4] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *Knowledge Discovery in Databases: PKDD 2007*, pages 506–514. Springer, 2007.

[5] D. Koller and M. Sahami. Toward optimal feature selection. 1996.

[6] J. Kreps, N. Narkhede, and J. Rao. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, 2011.

[7] J. Li, A. Ritter, and E. Hovy. Weakly supervised user profile extraction from twitter. ACL, 2014.

[8] M. Lipczak. Tag recommendation for folksonomies oriented towards individual users. *ECML PKDD discovery challenge*, 84, 2008.

[9] A. Mathes. Folksonomies-cooperative classification and communication through shared metadata. *Computer Mediated Communication*, 47(10):1–13, 2004.

[10] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in online social networks.

[11] R. Navigli and P. Velardi. Learning domain ontologies from document warehouses and dedicated web sites. *Computational Linguistics*, 30(2):151–179, 2004.

[12] I. Peters. *Folksonomies: Indexing and retrieval in Web 2.0*, volume 1. Walter de Gruyter, 2009.

[13] F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[14] I. Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[15] G. Salton and M. J. McGill. Introduction to modern information retrieval. 1986.

[16] B. Sigurbjörnsson and R. Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th international conference on World Wide Web*, pages 327–336. ACM, 2008.

[17] R. Sumbaly, J. Kreps, and S. Shah. The "big data" ecosystem at LinkedIn. In *SIGMOD '13*, pages 1125–1134, New York, NY, USA, 2013.

[18] L. Zhang, X. Wu, and Y. Yu. Emergent semantics from folksonomies: A quantitative study. In *Journal on Data Semantics VI*, pages 168–186. Springer, 2006.