# Combining Career Progression and Profile Matching in a Job Recommender System

Bradford Heap, Alfred Krzywicki, Wayne Wobcke, Mike Bain and Paul Compton

School of Computer Science and Engineering
University of New South Wales
Sydney NSW 2052, Australia
{bradfordh,alfredk,wobcke,mike,compton}@cse.unsw.edu.au

**Abstract.** In this paper we consider the problem of job recommendation, suggesting suitable jobs to users based on their profiles. We compare a baseline method treating users and jobs as documents, where suitability is measured using cosine similarity, with a model that incorporates job transitions trained on the career progressions of a set of users. We show that the job transition model outperforms cosine similarity. Furthermore, a cascaded system combining career transitions with cosine similarity generates more recommendations of a similar quality. The analysis is conducted by examining data from 2,400 LinkedIn users, and evaluated by determining how well the methods predict users' current positions from their profiles and previous position history.

## 1 Introduction

Searching for a new job is a difficult and time consuming process. The most common approach is for a user to enter search terms into a job search website. These search terms return a list of matched job advertisements which the user then self-evaluates [1]. While this method gives users fine control over finding potential jobs to apply for, users are unable to assess if they are amongst the most qualified or suitable for the desired job.

A job recommender is designed to match jobs to users, removing the need for manual search. The recommender should evaluate a user's suitability for jobs and recommend those that advance a user's career. Additionally, a recommender should ensure that jobs that are irrelevant, or for which the user is either over- or under-qualified, are *not* recommended. However, building a job recommender system using standard recommendation techniques is a challenging problem [9].

This work seeks to build a recommender that learns to predict career advancement from the job histories of other users. Our key contribution is the development of a recommender that is modular and uses a number of strategies to generate recommendations that promote career advancement. A core focus on this work is ensuring the developed system is able to be practically deployed and we do not suggest systems that could perform well but would have excessive processing costs. Through empirical evaluation we show that our recommender

system performs better than a baseline term-based recommender system.

As identified by Malinowski [7], job recommendation is characterized by bidirectional preferences, in that, while users have preferences for the jobs they desire, employers also have preferences for the candidates they interview or employ. Thus a job recommender needs to be a reciprocal recommender [8], taking into account the preferences of both parties. In this respect, job recommendation is similar to recommendation in online dating [3], however there are differences: (i) the turnaround time for job applications is typically much longer than in online dating, due to the need for employers to follow a well defined selection process, and (ii) employers may choose to interview only very few candidates out of many suitable, and typically employ only one. This results in an even greater sparsity of data available on the suitability of candidates for jobs. Data sparsity means that collaborative filtering, which works well in online dating settings [4], is more difficult to apply in job recommendation. Hence in this paper we focus on content-based recommendation algorithms for suggesting jobs to users based on their profiles and the job transitions of other users.

Datasets on candidates applying for jobs are difficult to obtain. In this work, we evaluate recommendation methods on a proxy problem, that of whether the method can identify a person's current position given their user profile, which includes their previous position history. We use data collected from 2,410 LinkedIn users (the one and two-step contacts of the authors and their colleagues). Being from LinkedIn, the dataset includes many IT professionals, however, also includes users from a variety of other fields, such as academia, healthcare, marketing, accounting, finance, management and HR. The data was collected over two distinct periods in 2013 and 2014 during which a number of users added new positions: users with these position titles were used as the test users, with their current position the target for prediction based on their earlier position history. Thus the recommenders make use only of information available in user profiles *before* their current position was commenced.

## 2 Research Motivation and Problem Description

Measuring the suitability of a user to a job is a difficult problem. In existing systems, it is common for similarity to be measured by matching keyterms extracted from previously viewed job advertisements [10, 1, 5] or a user's CV [7]. Both of these approaches to extracting keyterms have drawbacks. Using terms extracted from other job advertisements may be representative of a user's desires but it fails to consider the user's skills, qualifications and experience. In contrast, CVs generally consist of short summaries of a user's previous positions where the language used may differ vastly from the terminology used in job advertisements. For instance, a user may describe themselves as having "*a deep understanding of Object Oriented Programming principles*" where a job advertisement may be seeking someone with "*technical experience in Java/Groovy/Python application frameworks*". In this instance, as there are no terms in common, a simple term-based similarity measure would find no similarity between these two descriptions.

Sr. Recruiter (Hardware and Technology)

- Highly motivated Recruiter/Sourcer with 15 years of staffing **experience** who will generate a high volume/high **quality** pipeline of candidates
- Strong strategic and independent thinker ...
- Proficient with MS Office, recruiting databases and HRMS systems
- Specialties: Recruiting, Sourcing and Talent Acquisition Operating Systems: UNIX/HP-UX/Solaris/Linux, Windows NT/XP Languages: C/C++, Java, **XML**, **Javascript**, PHP, PERL, C#, Python, Ruby **Databases**: Oracle, **SQL** ...

Javascript Developer

An experienced native **Javascript** developer is required ...

Solid **experience** of the following technologies is needed:

- Strong programming skills in **JavaScript**
- HTML and XHTML, **XML**, CSS
- Working knowledge of relational **databases** and **SQL**
- Strong commitment to **quality** and customer success,
- Proven **experience** working with media and high traffic applications

Fig. 1: Example of Keyterm-Based Matching.

However, clearly there is some *semantic* similarity, and the user potentially has the skills for the advertised role. Furthermore, even in instances with many overlapping terms, poor quality recommendations can still be made if the *context* of terms is ignored. For instance, the skill of *networking* required for a sales manager is different from the skill required for a systems administrator.

An example that highlights a number of these issues is given in Fig 1. On the left is an excerpt from a user's profile, an HR professional who specialises in recruiting for IT jobs. On the right is a job advertisement for a programming role. Clearly this user is unsuitable for this role. Unfortunately, if we extract keyterms from both sets of free-form text (underlined terms) and then take the intersection of the two documents (bold terms) there are 8 common terms. Given that the job advertisement only has 21 extracted terms, using term-based profile matching the similarity between this user and the job advertisement is high. Examples like this are not uncommon, and highlight how difficult good term-based similarity matching can be in this domain. For many users these keyterm matches may be very good, especially if they work in the field of software development. However, this user is compromised by including many keyterms describing the detail of their work and few describing the higher level details of their role as a recruiter.

### 2.1 Problem Formalisation and Definitions

We now formalise the problem of recommending jobs to users. A *position* $p$ is held by a user $u$. Each position is a tuple $p = \langle T, D, s \rangle$ of a position title $T$, a *position description* $D$, and a *start date* $s$. Position titles and descriptions are treated as sets of keyterms $t$, $T = \{t_1, \ldots, t_m\}$, $D = \{t_1, \ldots, t_n\}$. A user's *position history* $H_u$ is a list of all positions currently or previously held ordered

by start date $H_u = [p_1, \ldots, p_o]$. All user profiles are contained in the set $U = \{H_1, \ldots, H_i\}$. A *job advertisement* is a tuple $j = \langle T, D \rangle$ of a position title $T$ and a position description $D$. A set of job advertisements for recommendation is $J = \{j_1, \ldots, j_i\}$. A *recommendation* $r$ is a tuple $r = \langle u, j, \sigma, f \rangle$ of a user $u$, a job advertisement $j$, a score $\sigma$, and a signal flag $f$. The purpose of the signal flag is to indicate the level of trust and distinctiveness in a recommendation. The signal flag can be set as strong or weak. Recommendations with strong signals are considered to be of higher value than those with weak signals.

### 2.2 Dataset Description

A user's profile on LinkedIn contains user authored free-form titles and summaries of current and former positions held. For our purposes, we ensure that each user profile in the dataset has at least two positions. To ensure positions in our dataset represent real jobs, we automatically validate each position title against a list of 5,599 distinct job titles, and remove any users with non-standard positions (e.g., company owners, founders, board members) as these jobs are not typically advertised. Finally, we ensure that each position description contains at least two distinct keyterms. In total, our dataset covers 2,410 candidate profiles consisting of 6,534 unique positions currently or previously held.

To extract keyterms from the user authored free-form text, we use the Apache OpenNLP library[1] to extract both single nouns and compound noun phrases. Using both single and compound noun extraction allows us to make partial matches on compound noun phrases. For instance, the compound term *software developer* will additionally be split into two single nouns, *software* and *developer*, allowing a partial match to the term *software engineer*. This approach allows us to generate more keyterms per profile allowing us to generate more potential matches for recommendation.

## 3 Recommender Construction

### 3.1 Baseline Term-based Similarity Recommender

We first develop a baseline term-based recommender. This recommender generates recommendations if there is overlap between the terms in a user's position history and a job advertisement. Similarity is measured using cosine similarity, treating both as documents, and is given as a score between 0 and 1 of how closely a user's position history matches a given job advertisement. The higher the value, the greater the number of common terms between the user's position history and the job advertisement relative to the number of terms in both.

An important feature of cosine similarity is the calculation of the normal of the length of both the set of user's keyterms and job advertisement keyterms, as this reduces the influence of user profiles with excessive keyterms generating matches with high scores (e.g., Fig. 1). Additionally, because this recommender

[1] https://opennlp.apache.org/

will generate a recommendation even if only one term is in common, many recommendations will be generated for each user.

Formally, let $\mathbb{D}$ be the set of all keyterms occurring in the titles and descriptions of positions in a user's position history $H_u \in U$ and $\mathbb{J} = T_j \cup D_j$ be all the the terms in the title and description of a job advertisement $j \in J$. The cosine recommendation score $\sigma_{u,j}$ of this user to the job advertisement is then:

$$\sigma_{u,j} = \frac{|\mathbb{D} \cap \mathbb{J}|}{\sqrt{|\mathbb{D}| \times |\mathbb{J}|}}$$

After the cosine similarity is calculated for all job advertisements, the recommender then returns a ranked list of recommendations $R = [r_1, \ldots, r_k]$ ordered by similarity score from highest to lowest. As there is no simple way to identify matches which are more distinctive than others, all recommendations made by this method are assigned a strong signal.

### 3.2 Career Transition Job Recommender

The career transition job recommender is designed as a cascaded system of recommendation modules, in which each module is an independent recommendation system (similar to Burke [2]). Each module explicitly generates recommendations of jobs for a user and each recommendation made contains a score which seeks to measure the strength of this recommendation. After recommendations are generated, a multiplexer is used to combine the recommendations made by each module and return a ranked list of final recommendations. Finally, if the number of recommendations made is below a preset threshold, the baseline cosine recommender may be called to supplement the number of recommendations produced. Fig. 2 is an overview of the modules and the cascaded design of the system.

In each module, a recommendation is flagged as a strong or weak signal relative to the other recommendations generated by the same module. Initially, any recommendation generated is considered to be a strong signal. However, if the total number of recommendations generated by a module, or a sub-component of a module, exceeds the preset *signal limit* then all recommendations generated by that module, or sub-component, are flagged as a weak signal. This identifies modules generating many low quality recommendations for a particular user.


Fig. 2: Cascaded Career Transition Job Recommender.

Table 1: Excerpt of Title Transition Terms for *Software Developer*.

| Previous title term | Transition title terms |
|---|---|
| software developer | senior, programmer, analyst, analytics, developer |
| software | engineer, developer, senior, technical, specialist |
| developer | engineer, senior, software, specialist, ios |

After a module generates a set of recommendations, they are sent to the multiplexer. The multiplexer merges together the sets of recommendations from each module. If equivalent recommendation pairs (that is, where $\langle u, j\rangle \in r = \langle u, j\rangle \in r'$) are generated in different modules, they are aggregated into a single recommendation. To aggregate equivalent recommendations, the scores from each individual recommendation are added together and the maximum strength of the signal flags is used to form a final recommendation. The multiplexer only allows recommendations with strong signals to be returned, except that after aggregation, any weak signal recommendations with scores that exceed the largest strong signal recommendation score are promoted to strong signal recommendations. This ensures that any recommendations that have been generated as weak signals by multiple modules but have a large cumulative score are also returned.

We now describe each of our recommendation modules:

**Title Transition Module** This module contains two sub-components which evaluate the similarity of an advertised job title to a user's current and previous job title terms. The first of these sub-components, *term consistency*, seeks to maintain the status quo and considers each keyterm in a user's job title and assumes that this remains the same in a new job. Formally, let $\mathbb{T}$ be the set of all key terms occurring in the position titles in a user's position history. The terms in common with an advertised job title $T_{j\in J}$ is the set $\mathbb{T} \cap T_j$.

The second component, *term transition*, uses a table built from the set of all user position histories $U$ to suggest new title terms based on the user's current and previous title terms. The rows of the transition table are pairs of title terms $\langle t_{\in T_{p_i}}, t'_{\in T_{p_{i+1}}}\rangle$ where $p_i, p_{i+1}$ are successive positions in a user's position history $H_u$. To avoid spurious term transitions, we ensure at least two different position transitions support the same term transition pair. Every title term in a user's position history $t \in \mathbb{T}$ is then used to generate the set of transition terms $T_s$ which is then combined with the terms from an advertised job title to find the similarity $T_s \cap T_j$.

Using individual keyterms in this module, rather than full titles, allows both the term consistency and term transition sub-components to generate many possible job recommendations. For instance, a user with the job title *software developer* will be recommended jobs by the term consistency sub-component that contain either *software* or *developer* in their title. The term transition sub-component will then suggest other possible job title terms. Table 1 gives an excerpt of these transition terms. This gives us an insight into how title terms

transition between roles. For instance, we note that all three terms have transitions to the new term *senior*. This suggests that users who have the role of a software developer commonly transition into a job with *senior* in the title. Additionally, this table shows a close relationship between the terms *software* and *developer*, as the two terms are suggested as transitions from one to the other.

**Title Description Matching Module** This module assumes that terms appearing in the titles of a user's previous positions influence their next position description. We assume that words in a user's previous position titles are key descriptors of their roles and it is common for future roles to have some overlap in responsibilities with past roles. Consider a user whose previous position titles are *business analyst* and *systems architect* who applies for the position of *project manager*. Despite no terms in common between these position titles, it can be expected that the position description for the *project manager* role would include the words business and systems.

For this module to generate a recommendation, there must be at least two terms in common. The terms in common between the title terms in a user's position history and a job description is the intersection $\mathbb{T} \cap D_j$. In general, it can be expected that this module will generate similar recommendations to the title transition module. However, unlike the title transition module which uses 1-to-1 term matching to make recommendations, through the merging of all title terms in a user's position history, this module combines information from a number of previous job titles to make recommendations.

**Description-Description Matching Module** The third module measures the similarity between the description terms in a user's position history and a job description. The core of this module is the same as the baseline cosine similarity recommender. However, the recommendation scores are limited to a maximum similarity score. As this module generates a very high number of recommendations, the recommendations are generally flagged as weak signals and if they are only generated by this module they discarded by the multiplexer. More commonly, the scores assigned to recommendations generated by this module are aggregated by the multiplexer with recommendations generated in the other modules to boost strong signal recommendation scores.

**Module Recommendation Scoring and Signal Limits** Each module is responsible for scoring their generated recommendations relative to their own assessment criteria – generally as a proportion of the number of terms in common. The higher the score, the more relevant the module determines this recommendation to be. Each module and sub-component is limited to a maximum score it can assign to any individual recommendation. Table 2 shows the maximum scores for each module and sub-component in our system. We set these scores according to the ratio of the total number of recommendations generated and the total correct recommendations made by each module (in future work these

Table 2: Module Scoring

| Module | Max Score | Signal Limit |
|---|---|---|
| Title Transition: Term Consistency | 250 | 20% |
| Title Transition: Term Transition | 100 | 20% |
| Title Description Matching | 200 | 10% |
| Description-Description Matching | 800 | 10% |
| Cosine Supplementation | 150 | - |

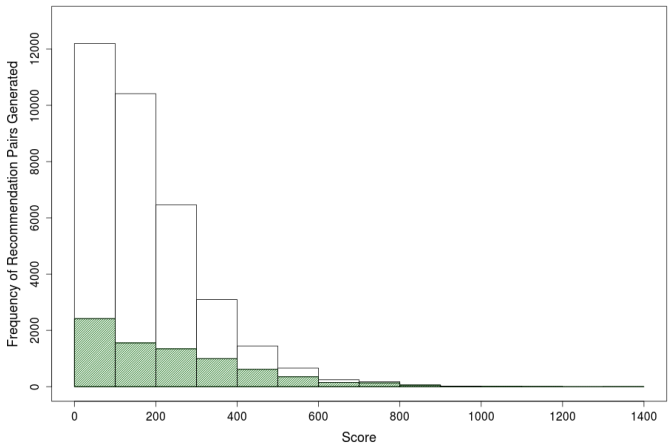


Fig. 3: Effect of Signal Limit on Low Score Recommendations.

scores could be learned). This table also shows the signal limit for each module. The signal limit is a proportion of all possible recommendations that can be generated.

Fig. 3 shows the effect of signal limitation. The shaded area of the histogram shows the frequency of scores from recommendations made after the application of the signal limit. The unshaded frequencies show the number of recommendations and distributions of scores without any signal limitation. This demonstrates clearly that the application of the signal limit prevents many recommendations with low scores from being made. This ensures that recommendations in the final set of recommendations made have a high level of distinctiveness.

## 4 Experimental Evaluation

We measure the relative performance of our career transition job recommender against the baseline term-based recommender using 10x2-fold cross-validation. It should be noted that these recommenders are measured against the *positive* ground truth only, that is, where there is one exact match for each user/job recommendation pair, and the goal of recommendation is to cover as many of these

Table 3: Recommender Results for 10x2-fold Cross-validation.

| | All Recommendations | | Top-$N$ Recommendations | | | |
|---|---|---|---|---|---|---|
| | Generated | Recall | $N = 5$ | $N = 10$ | $N = 20$ | $N = 40$ |
| Cosine Baseline | 49.9% | 79.4% | 27.1% | 31.3% | 36.8% | 43.0% |
| Career Transition | 13.3% | 52.1% | 29.5% | 34.0% | 38.3% | 42.8% |
| Career Transition + Cosine | 30.6% | 65.8% | 29.5% | 34.1% | 38.5% | 43.3% |

pairs as possible. This is because it is not established that a user would definitely not be interested in alternative jobs if they were recommended, and hence such recommendations should not be regarded as false positives (in fact, such jobs are often quite similar to those in the ground truth pair). Therefore the core metric that we use to measure the performance of our recommenders is recall. Recall is the proportion of user-position pairs from the test dataset predicted correctly by recommendations. Using the ranking of recommendations according to their similarity score, we report recall for the top $N$ recommendations for various values of $N$.

### 4.1 Results

Table 3 shows the mean results for each recommender shown as percentages out of the largest possible result. These results show that the baseline recommender generates a large number of recommendations, nearly half of all possible recommendation pairs that can be generated. Consequently, this results in large recall. In contrast, the career transition job recommender generates only one fourth as many recommendations but maintains a high level of recall. When the cosine recommender is used to supplement the number of recommendations given by the career transition job recommender, recall increases. Across all three recommenders, almost all users receive recommendations and nearly all jobs are recommended to at least one user.

More pertinent is the recall when the recommendations are limited to the top $N$ by score. Table 3 and Fig. 4 show the increase in recall as $N$ increases. The first plot shows that when the number of recommendations is limited to $N < 40$ the career transition recommender with cosine supplementation has a higher level of recall than the baseline recommender. This is a strong result, as being able to make accurate recommendations is an important feature of a job recommender system. The second plot shows the benefit of cosine supplementation when $N$ is large. However, for values of $N < 20$ there is no difference between the two recommenders.

Finally, using paired-sample t-tests we compared the recall results for each recommender at various levels of $N$. The results showed a significant difference ($P < 0.01$) in recall for the career transition job recommender compared to the baseline recommender.

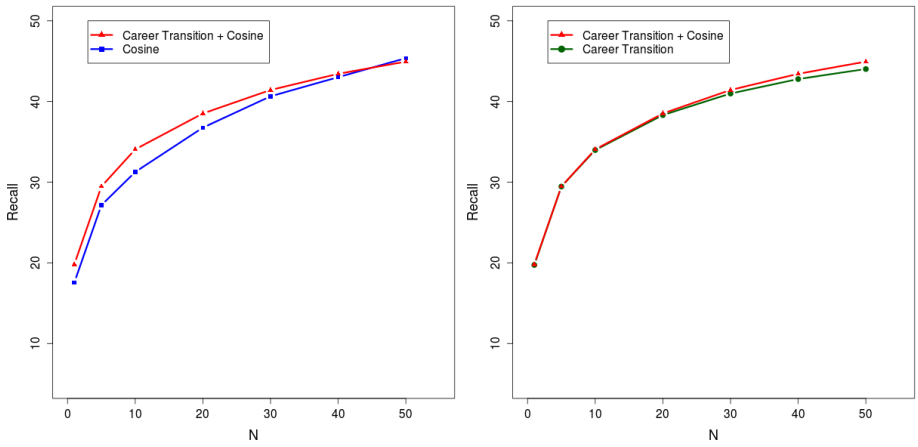


Fig. 4: Recall for Top-$N$ Recommendations.

### 4.2 Module and Sub-component Evaluation

Table 4 breaks down the performance of the recommender by modules and module combinations. These results provide insight into how each component contributes recommendations, recall, and ranking to the overall system.

The first section of this table breaks down the recommender by module. The results for the title transition module show that for each user, on average, 10% of the job advertisements are recommended and for 4 in every 10 users one recommendation will be correct. This is a high level of recall for a relatively small number of recommendations generated. The second module, title description matching, produces even fewer recommendations per user but has an impressive recall for so few recommendations. In contrast, the description-description module produces a very large number of recommendations and, despite having the largest recall, the ratio of recommendations to recall is small.

The second section of Table 4 explores combinations of modules. It shows that the combination of the title transition module and the title description matching module generates more recommendations and higher recall than the two modules in isolation. This suggests that the individual modules are generating different recommendations, consequently the top-$N$ recall results for this module also show improvement after the combination of these two modules.

The second combination of title transtion with description-description matching shows a large reduction in the number of recommendations generated by the description-description matching module in isolation as a consequence of the signal limit being applied when combining recommendations. Despite this, the top-$N$ recall for the combinations of these modules is an increase over their individual performance. This again shows that despite producing fewer recommendations than the individual description-description matching module, the rec-

Table 4: Module and Module Combinations.

| | All Recommendations | | Top-$N$ Recommendations | | | |
|---|---|---|---|---|---|---|
| | Generated | Recall | $N = 5$ | $N = 10$ | $N = 20$ | $N = 40$ |
| **Individual Modules:** | | | | | | |
| Title Transition (TT) | 10.0% | 40.9% | 15.4% | 19.4% | 24.0% | 29.2% |
| Title Description (TDM) | 2.9% | 17.0% | 5.4% | 7.6% | 10.4% | 13.0% |
| Description-Description (DDM) | 50.0% | 79.3% | 26.2% | 31.1% | 36.6% | 43.0% |
| **Module Combinations:** | | | | | | |
| TT + TDM | 11.1% | 44.0% | 16.9% | 21.2% | 26.0% | 31.0% |
| TT + DDM | 12.4% | 50.7% | 28.7% | 33.3% | 37.5% | 41.7% |
| TDM + DDM | 17.1% | 50.7% | 26.4% | 30.3% | 35.3% | 39.0% |
| **Overall:** | | | | | | |
| Career Transition | 13.3% | 52.1% | 29.5% | 34.0% | 38.3% | 42.8% |
| Career Transition + Cosine | 30.6% | 65.8% | 29.5% | 34.1% | 38.5% | 43.3% |

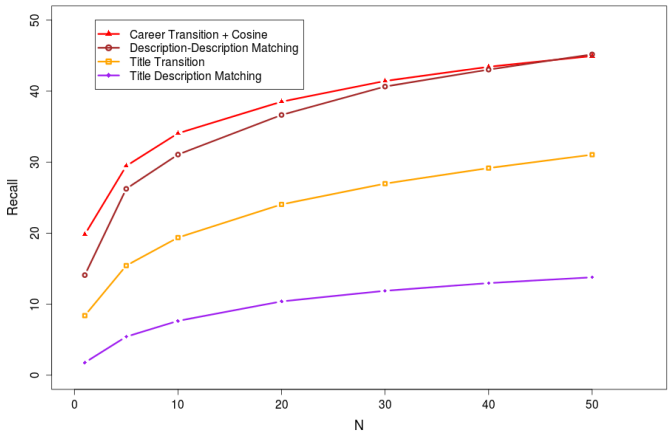


Fig. 5: Recall for Top-$N$ Recommendations by Module.

ommendations that are retained after multiplexing are of high value. The third combination, title description matching with description-description matching, also shows a decrease in the number of recommendations generated and the same level of recall as the previous combination. However, when considering top-$N$ recall, there is a minor reduction compared to the individual results for the description-description matching module.

Finally, the third section of the table shows the combination of all three modules into the final recommender and the addition of cosine supplementation on the final results. Fig. 5 shows the individual results for each module and how the combination of all three is higher than each in isolation.

## 5 Related Work

Two deployed job recommender systems are CASPER [1] and Proactive [5]. CASPER is a two-stage search engine based recommendation system. First, similarity metrics are used to find job advertisements that are relevant to a candidate's search term. Second, a personalised ranking of the returned search results is made based upon the candidate's learned profile which includes prior likes and dislikes for similar jobs. This enables two candidates who input identical search terms to see the returned search results in a different personalised order.

Learned candidate profiles are central to CASPER's personalisation service. Each profile is made up of metadata (e.g., location, salary, skills) from the job ads that the user has previous viewed. Similarity between the learned user profile and search results is then measured through a nearest neighbour classifier. An evaluation of different similarity metrics has been made through generated artificial candidate profiles [1].

The candidate profiles used for recommendation in the Proactive system are quite different from CASPER. Candidates using Proactive build a collection of *liked* jobs. These liked jobs are then matched to job advertisements to generate a list of recommended jobs. To match jobs, Proactive extracts metadata from job descriptions and uses seven *job case facets* (e.g., role, industry, company size, location, qualifications, experience) to determine a similarity (again using a nearest-neighbour classifier) to a job for recommendation. This approach allows candidates to control which recommendations are generated based on their direct action of liking a particular job. Unfortunately, the Proactive system has been limited to IT jobs and evaluated on a very small set of users [6].

While both CASPER and Proactive demonstrate that term-based profile matching can be used to filter or display relevant job ads to candidates, it remains a relatively crude approach. Recommendations given by either system may be irrelevant (in the case of CASPER's learned profiles) or of poor quality, for instance, in situations where the candidate may have a high desire for a particular job but would be an unsuitable employee.

## 6 Conclusion and Future Work

In this paper we have considered the problem of job recommendation, suggesting suitable jobs to users based on their profiles. We considered a baseline method treating users and jobs as documents, where suitability is measured using cosine similarity, and a model that incorporates job transitions trained on the career progressions of a set of users. We showed that the job transition model outperforms cosine similarity. Furthermore, a cascaded system combining career transitions with cosine similarity generates a larger number of recommendations of a similar quality.

Given a better dataset, our recommendation methods could be extended by including location-based job filtering, using temporal information associated with the length of time a user has held a certain job level (c.f. Wang *et al.* [11]), and incorporating users' qualifications and skills.

## Acknowledgements

This work was funded by Smart Services Cooperative Research Centre. We would like to thank Infosys, especially Jai Ganesh, for their support of this research.

## References

1. Bradley, K., Smyth, B.: Personalized Information Ordering: A Case Study in Online Recruitment. Knowledge-Based Systems 16, 269–275 (2003)
2. Burke, R.: Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12, 331–370 (2002)
3. Cai, X., Bain, M., Krzywicki, A., Wobcke, W., Kim, Y.S., Compton, P., Mahidadia, A.: Collaborative Filtering for People to People Recommendation in Social Networks. In: Li, J. (ed.) AI 2010: Advances in Artificial Intelligence. Springer-Verlag, Berlin (2010)
4. Krzywicki, A., Wobcke, W., Cai, X., Mahidadia, A., Bain, M., Compton, P., Kim, Y.S.: Interaction-Based Collaborative Filtering Methods for Recommendation in Online Dating. In: Chen, L., Triantafillou, P., Suel, T. (eds.) Web Information Systems Engineering – WISE 2010. Springer-Verlag, Berlin (2010)
5. Lee, D.H., Brusilovsky, P.: Fighting Information Overflow with Personalized Comprehensive Information Access: A Proactive Job Recommender. In: Proceedings of the Third International Conference on Autonomic and Autonomous Systems, p. 21 (2007)
6. Lee, D.H., Brusilovsky, P.: Proactive: Comprehensive Access to Job Information. Journal of Information Processing Systems 8, 721–738 (2012)
7. Malinowski, J., Keim, T., Wendt, O., Weitzel, T.: Matching People and Jobs: A Bilateral Recommendation Approach. In: Proceedings of the 39th Hawaii International Conference on System Sciences (2006)
8. Pizzato, L., Rej, T., Akehurst, J., Koprinska, I., Yacef, K., Kay, J.: Recommending People to People: The Nature of Reciprocal Recommenders with a Case Study in Online Dating. User Modeling and User-Adapted Interaction, 23, 447–488 (2013)
9. Rafter, R., Bradley, K., Smyth, B.: Automated Collaborative Filtering Applications for Online Recruitment Services. In: Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 363–368 (2000)
10. Rafter, R., Bradley, K., Smyth, B.: Personalised Retrieval for Online Recruitment Services. In: Proceedings of the 22nd Annual Colloquium on Information Retrieval (2000)
11. Wang, J., Zhang, Y., Posse, C., Bhasin, A.: Is It Time for a Career Switch? In: Proceedings of the 22nd International World Wide Web Conference, pp. 1377–1388 (2013)