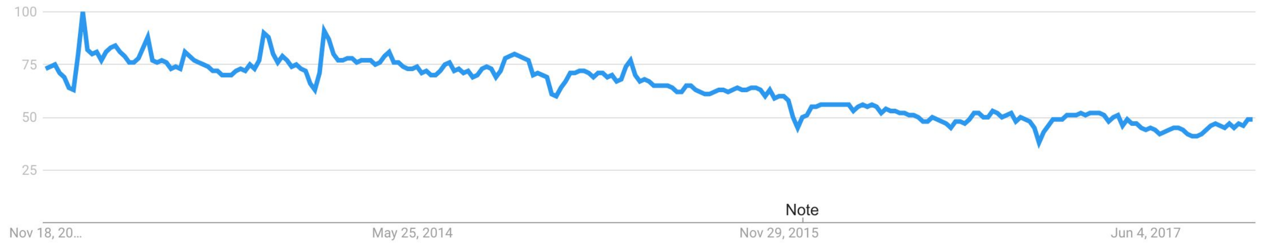
On utility of temporal embeddings for skill matching

Manisha Verma, PhD student, UCL
Nathan Francis, NJFSearch

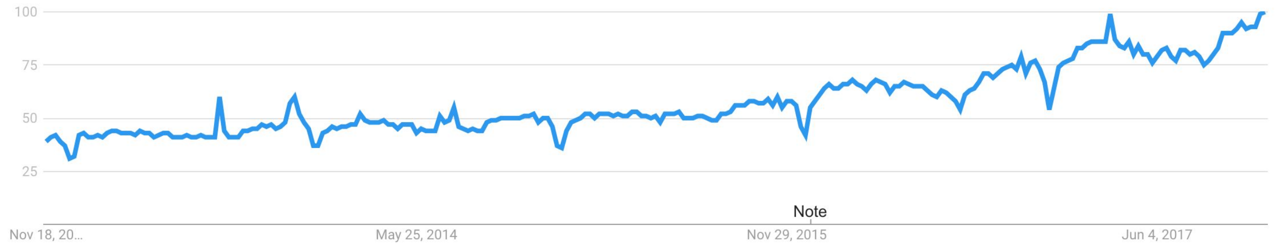
Skill Trend Importance

1. Constant evolution of labor market yields differences in importance of skills.

JAVA

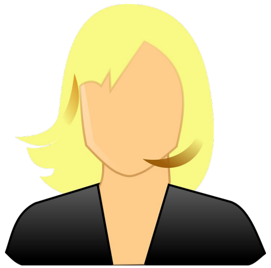


Python

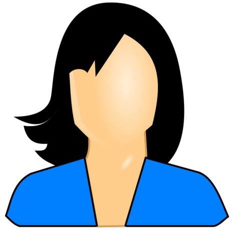


Skill Trend Importance

2. Candidate resumes have myriad number of skills.



Information Retri… · 46	Distributed Systems · 42	Scalability · 32
Big Data · 22	Software Enginee… · 22	MapReduce · 21
Computer Science · 19	C++ · 18	Data Mining · 17
Python · 17	Unix · 16	Text Mining · 15

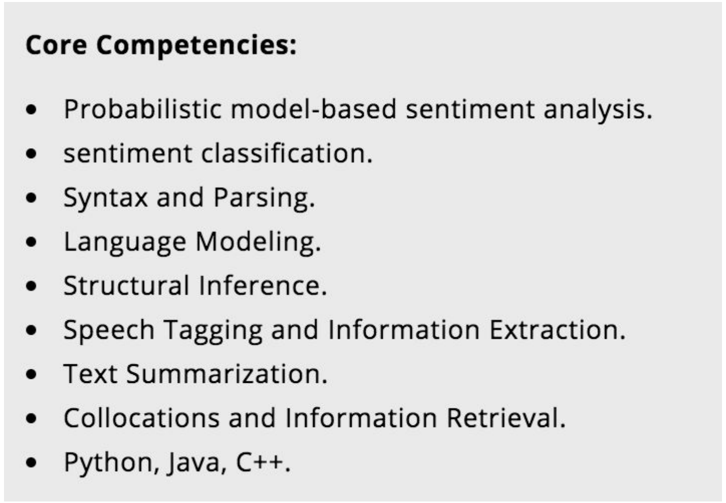


Algorithms · 79	Computer Science · 75	Artificial Intellige… · 50
Computational Li… · 30	Data Mining · 30	Pattern Recognition · 20
Web Mining · 19	Text Mining · 16	Object Oriented D… · 11
Text Classification · 8	MapReduce · 6	Recommender Sy… · 3

Skill Trend Importance

3. Candidate-job matching relies on candidate-skill scoring

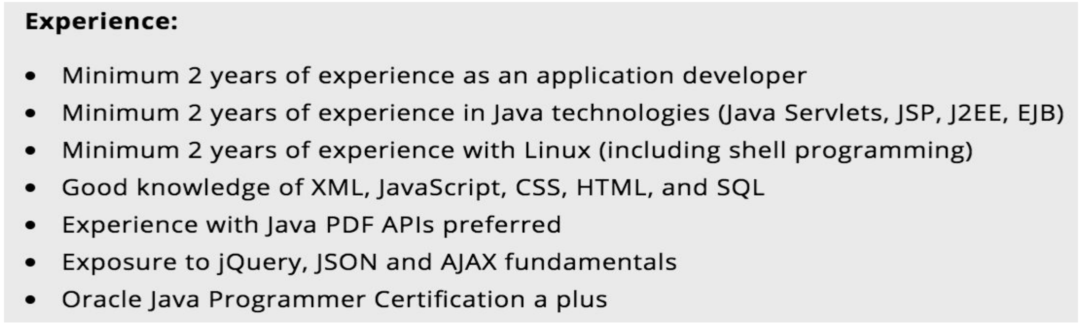
JOB A: NLP Scientist



Core Competencies:

- Probabilistic model-based sentiment analysis.
- sentiment classification.
- Syntax and Parsing.
- Language Modeling.
- Structural Inference.
- Speech Tagging and Information Extraction.
- Text Summarization.
- Collocations and Information Retrieval.
- Python, Java, C++.

JOB B: Java developer



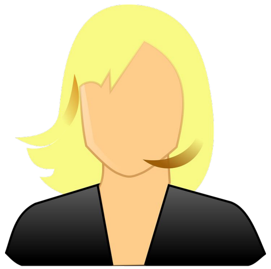
Experience:

- Minimum 2 years of experience as an application developer
- Minimum 2 years of experience in Java technologies (Java Servlets, JSP, J2EE, EJB)
- Minimum 2 years of experience with Linux (including shell programming)
- Good knowledge of XML, JavaScript, CSS, HTML, and SQL
- Experience with Java PDF APIs preferred
- Exposure to jQuery, JSON and AJAX fundamentals
- Oracle Java Programmer Certification a plus

Problem Statement

1. Determine temporal importance or utility of a skill.
2. Match candidates to skills by taking into account skill's temporal importance.
3. Suggest temporally relevant and related skills with respect to an input skill.

Problem Statement



Information Retri… · 46	Distributed Systems · 42	Scalability · 32
Big Data · 22	Software Enginee… · 22	MapReduce · 21
Computer Science · 19	C++ · 18	Data Mining · 17
Python · 17	Unix · 16	Text Mining · 15

Temporal Relevance

Python · 17   Text Mining · 15   Data Mining · 17   MapReduce · 21   C++ · 18   Unix · 16

Related Work

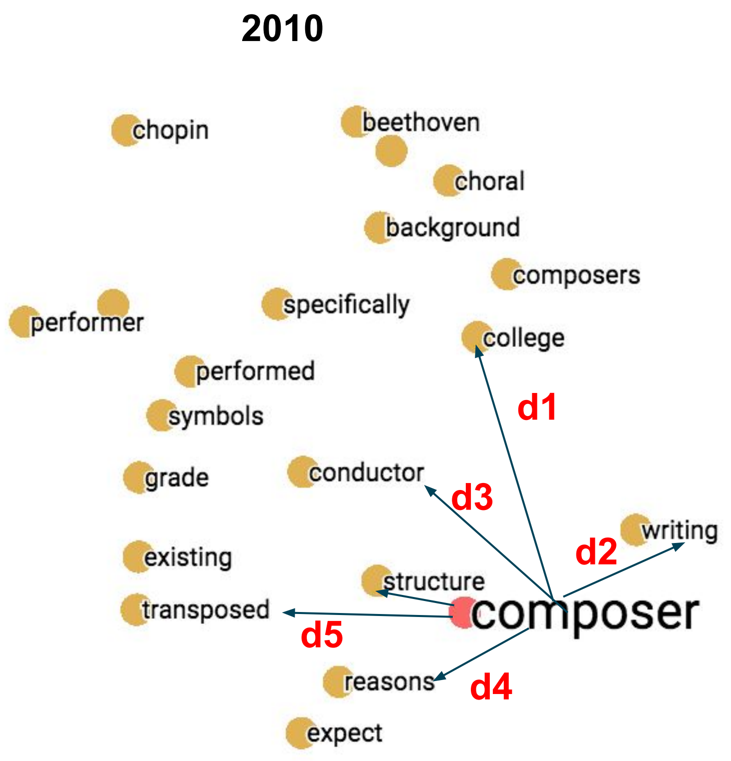
1. Existing work is limited to ranking jobs or recommending candidates on basis lexical or semantic similarity.
2. Skills, jobs or candidate profiles are represented using k-dimensional vectors determined using bag-of-words, graph algorithms, topic modelling or word embeddings.

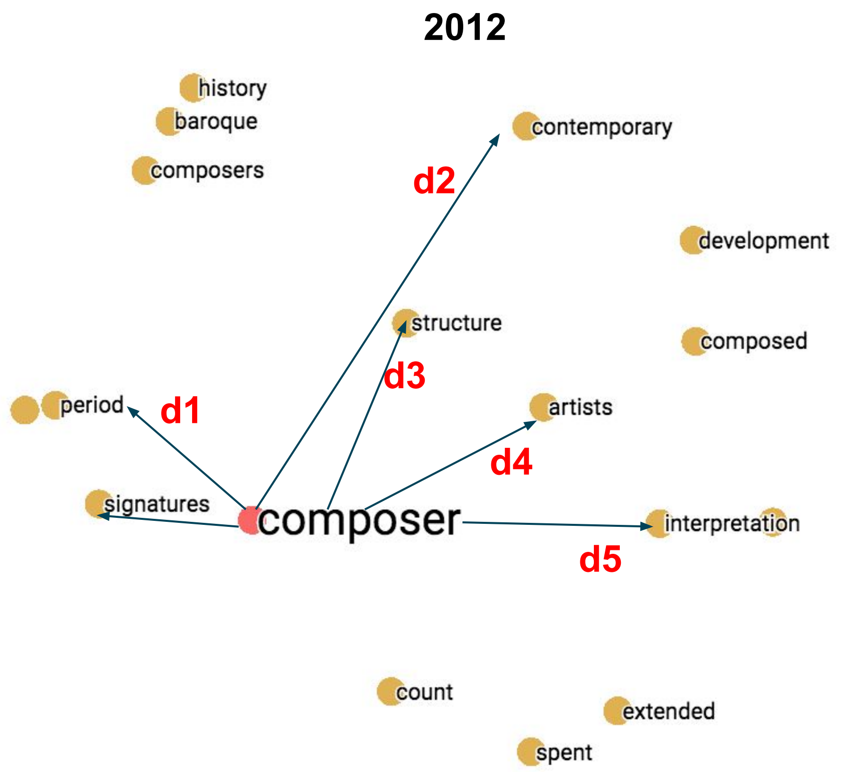
Temporal information is missing in skill, candidate or job vectors!!!

Proposed Solution

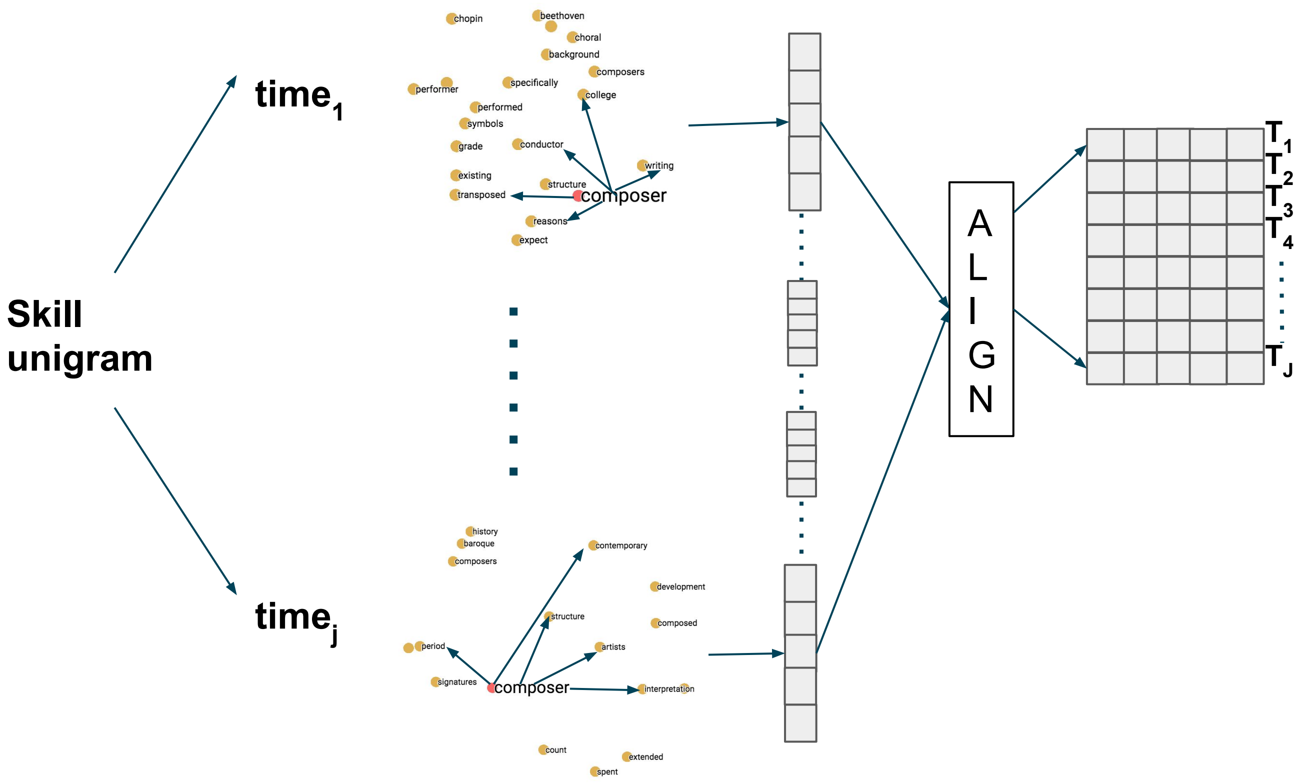
1. Exploit word embeddings.
2. Construct embeddings for different time periods.
3. Align these temporal embeddings to compare keywords across different time periods.
4. A keyphrase of skill is represented using multiple embeddings spanning a large time frame.

Proposed Solution





Proposed Solution



Each skill is now represented using a 2-D tensor.

Bi-gram or Tri-gram tensors can be computed by aggregating (sum or average) individual unigram tensors.

Temporal Embeddings

Vector representations of skills are constructed using two approaches:

SVD: It is low-rank approximation of data. SVD representations are robust as dimensionality reduction ensures regularization.

Skip-gram with negative sampling or Word2Vec

- Embed unigrams in lower dimension using neural network based language models.
- Predicts co-occurrence of words with an approximate objective of skip-gram with negative sampling.
- Each word $\boldsymbol{w_i}$ is represented using two dense but low-dimensional vectors: a word vector $\boldsymbol{v_i^{sgns}}$ and a $\boldsymbol{c_i^{sgns}}$ context vector that represents its neighbourhood.

Embedding Alignment

With W(t) ∈ $R^{|V|.d}$ as the matrix of word embeddings learned for year *t*, align embeddings across time-periods while preserving cosine similarities between pair of embedding matrices spanning two time periods by optimizing the following objective function:

$$R(t) = \underset{Q^T Q = I}{\arg\min} ||Q \cdot W(t) - W(t+1)||_2$$

with R(t) ∈ R^{dxd} , where || · || represents the Frobenius norm.

Skill Scoring

HYPOTHESIS: Skills popular at time ***t*** should get preference over skills popular at time ***t* − *δ***.

Adopt discounted importance to score skills.

Captures the **underlying trend** of keyword **across multiple time frames**. We represent a token $\boldsymbol{w_i}$ at time ***t***, with temporally aligned embedding $\mathbf{v}_{it} \in \mathbf{R}^d$. Similarity between two skills $\mathbf{s}_i$ and $\mathbf{s}_j$ is given by:

$$sim(s_i, s_j) = \sum_t 2^{t\lambda} \cdot \frac{\frac{1}{k}\sum_{k\in|s_i|} v_{kt} \cdot \frac{1}{l}\sum_{l\in|s_j|} v_{lt}}{||\frac{1}{k}\sum_{k\in|s_i|} v_{kt}||_2 \quad ||\frac{1}{l}\sum_{l\in|s_j|} v_{lt}||_2}$$

Dataset

1. **Stack-exchange** question answering dataset to compute temporal skill representations for computer science. In the absence of publicly available large scale time oriented skill dataset, we use this dataset as a proxy.
2. The dataset contains **10 million questions, 6000 tags and 1 Million unique tokens**.
3. Use the **body, title and tags** of the stackexchange posts to construct these embeddings.
4. We train embeddings for five years between **2012-2017**.

Dataset

Skill Matching

1. We use dataset of 100 skills from 2000 computer science candidates profiles across multiple websites.
2. For each skill, manual evaluation of related skills was conducted. 10.7 skills were evaluated on average against each skill

Candidate Matching

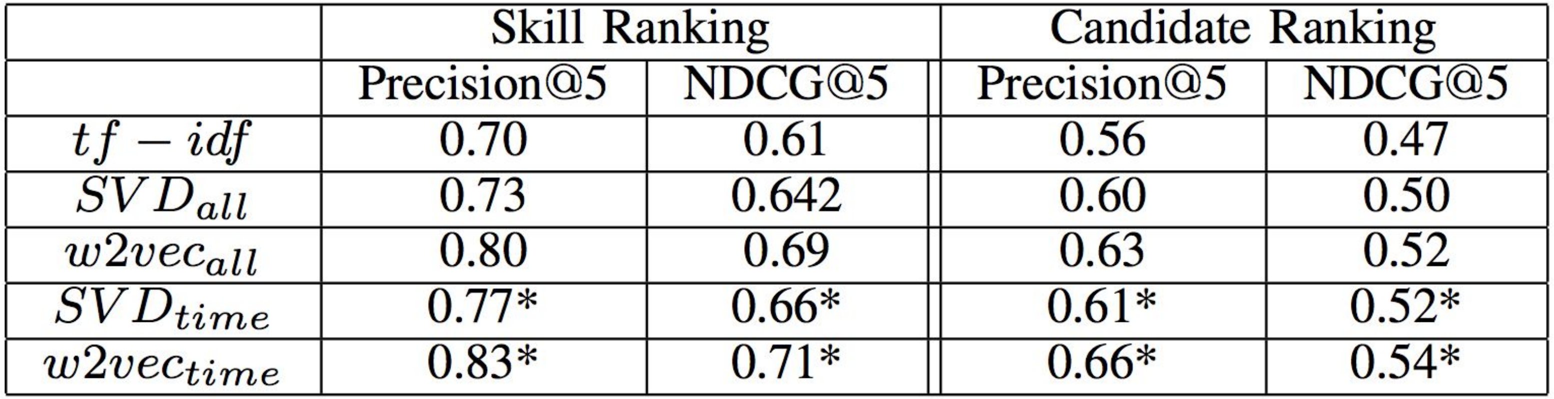
1. Collected labels for 30 skills and 70 candidate profiles in computer science. Ground truth consists of average 7.3 skills per candidate.
2. For each candidate profile, the annotator was shown a ranked list of related skills extracted from candidate's profiles.

Baselines and Metrics

1. **Tf-idf** : A simple baseline that takes into account the ratio of skill popularity in the candidate profile and its frequency in dataset.
2. **SVD$_{\textbf{all}}$**: Vector representations built using entire corpus of candidate profiles by using Singular valued decomposition.
3. **word2vec$_{\textbf{al}}$**: Skill vector representations constructed from all candidate profiles using word2vec.
4. **SVD$_{\textbf{time}}$**: Time based representations of skill unigrams using SVD.
5. **word2vec$_{\textbf{time}}$**: Time based representations of skill unigrams using word2vec.

Results

Report **Precision** and **NDCG** as we evaluate ranking accuracy



	Skill Ranking		Candidate Ranking	
	Precision@5	NDCG@5	Precision@5	NDCG@5
$tf - idf$	0.70	0.61	0.56	0.47
SVD_{all}	0.73	0.642	0.60	0.50
$w2vec_{all}$	0.80	0.69	0.63	0.52
SVD_{time}	0.77*	0.66*	0.61*	0.52*
$w2vec_{time}$	0.83*	0.71*	0.66*	0.54*

Limitations

1. The study is conducted on a small dataset
    a. Large scale job descriptions and candidate profiles would yield significantly different temporal embeddings.
2. The scoring function is very simple.
    a. A more refined approach would be to implement learning-to-rank functions that encode ***both relevance and temporal importance***.

Future Work

1. Temporal job recommendation
   a. With temporal importance of skills, rank temporally relevant jobs wrt to candidate profile.
2. Skill importance prediction
   a. On basis of existing career trajectories, predict temporal relevance of skill for particular candidate/job market