Evaluation of Filtering Methods Applied to the Unstructured Datasets in the Predictive Learning Services

Dmitry Ilin[1,2], *Egor* Mateshuk[3,*], *Rustam* Gilaztdinov[2], and *Gregory* Bubnov[1,3]

[1]Moscow Technological Institute, 119334, Moscow, Russia
[2]Moscow Technological University MIREA, 107996, Moscow, Russia
[3]Moscow Institute of Physics and Technology, 141700, Dolgoprudny, Moscow Region, Russia

Abstract. Predictive learning services perform aggregation and homogenization of open data from public sources, in particular from the online recruitment agencies. However, the sample of vacancies may contain various percentage of noise due to the frequent occurrence of homonyms. This article will consider two approaches of noise reduction: the first one is based on the cosine similarity and the second one is based on the contextual words.

1 Introduction

Many of the major online services provide application programming interface (API) for integration of third-party applications, particularly adhering to the REST principles [1]. Some of the online recruitment agencies (e.g. Indeed.com [2], HeadHunter [3]) provide an API for fetching a vacancy list matching the specified search criteria.

During the development of the predictive learning service [4], the vacancy data of the IT sector were obtained from the mentioned agencies in order to analyze which skills are the most demanded and which are the fastest-growing in demand. However, in the course of experiments it was found that the data from the Indeed.com may comprise a significant percentage of noise. One of the reasons is unstructured vacancy descriptions. For example, a sample of vacancies with the Haskell programming language requirement had noise percentage about 54%, which means the unsuitability of data to assess the real needs of the market in regard of the corresponding skills.

Table 1. Online recruitment agencies' API limitations

	Indeed	**HeadHunter**
Maximal number of documents in the sample	1025	2000
Maximal document size	~200 characters	~400 characters
Historical data	No	No

The binary text classification is well studied [5, 6], but it hasn't been applied to the analysis of the demand for skills yet. In addition, the integration of the filtering into the predictive learning service, with regard to the existing constraints (Table 1), is a relevant issue.

2 Materials and Methods

2.1. Materials

Samples of vacancies taken from the Indeed and HeadHunter (Tables 2 and 3 respectively) were processed manually and classified according to the following rule: job title or description must clearly specify requirement of the skill. In other words, the vacancies containing the text of a technical orientation, but not containing an explicit reference to skill/technology, are considered noise. The use of keywords within the meaning other than the skill is considered to be noise as well.

Possible error in the table values – 5 vacancies which does not affect the result. Haskell, Boo and Clarion skills were excluded from Russian online recruitment agency data, due to the fact that the number of the results was lower than the possible error.

Figures 1 and 2 show visualisation of the vacancy to noise ratio.

Significantly lower noise percentage in the second sample due to the fact that in the HeadHunter service, where the main language is Russian, the probability of the use of professional English terms with varying semantics is lower than in the Indeed, where the main language is English.

* Corresponding author: e.mateshuk@gmail.com

Table 2. Vacancy to noise ratio in the data sample from the Indeed service.

Skill	Total vacancies	Relevant vacancies	Noise vacancies
Java	1025	945	80
PHP	1025	897	128
Objective-C	1025	379	646
Haskell	373	171	202
Boo	88	2	86
Pascal	67	37	30
MySQL	1025	784	241
Microsoft Access	1025	433	592
SQLite	471	320	151
Clarion	412	24	388

Table 3. Vacancy to noise ratio in the data sample from the HeadHunter service.

Skill	Total vacancies	Relevant vacancies	Noise vacancies
Java	484	448	36
PHP	416	386	30
Objective-C	53	50	3
Pascal	8	8	0
MySQL	346	298	48
Microsoft Access	20	18	2
SQLite	14	14	0

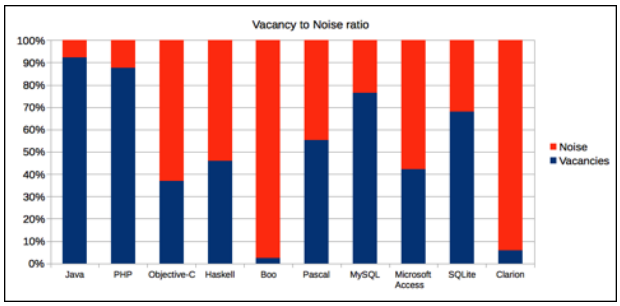

Fig. 1. Vacancy to noise ratio in the data sample from the Indeed service.

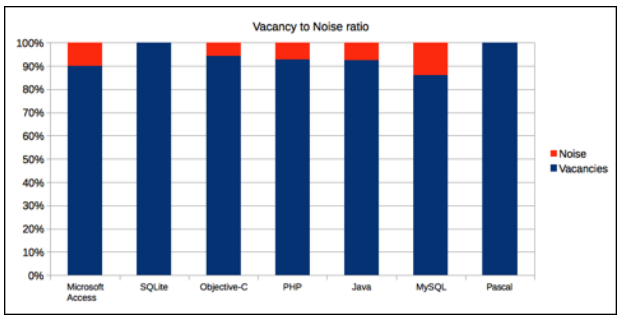

Fig. 2. Vacancy to noise ratio in the data sample from the HeadHunter service.

2.2 Methods

Due to the sample limitations, word2vec [7] and doc2vec [8] models perform with low efficiency. Therefore two different approaches have been chosen as the possible filtration method:

1. Unsupervised, based on the use of cosine similarity.
2. Supervised, based on the use of contextual words.

The first approach is based on the calculation of the cosine similarity between the vector representations of the vacancy texts. The algorithm consists of the following steps:

1. Tokenization.
2. Stemming.
3. Removal of the stop-words.
4. Calculation of the cosine similarity matrix between all the vacancies.
5. Calculation of an ordered list of factors.
6. Selection of the reference factor.
7. Filtering based on the reference factor and standard deviation.

The algorithm is shown in the form of a pseudocode in Figure 3.

```
CosineSimilarityFilter(vacancyTextList)

 tokenOccurencesList = []
 for vacancyText in vacancyTextList
  rawTokenList <- Tokenize(vacancyText)
  stemmedTokenList <- PorterStemming(rawTokenList)
  tokenList <- RemoveStopWords(stemmedTokenList)
  tokenOccurencesMap <- CountWordOccurences(tokenList)
  tokenOccurencesList += tokenOccurencesMap

 csMatrix = [][]
 for to1, i in tokenOccurencesList
  for to2, j in tokenOccurencesList
   csMatrix[i][j] <- CosineSimilarity(to1, to2)

 similarityList = []
 for csList in csMatrix
  similarityList += Mean(csList) + 0.75 * Std(csList)
 similarityList <- Sort(similarityList, 'desc')

 referenceValue <- First(similarityList)
 lBound <- referenceValue - 3 * Std(similarityList)
 uBound <- referenceValue + 3 * Std(similarityList)

 filteredVacancyList = []
 for similarity in similarityList
  if (similarity > lBound AND similarity < uBound)
   filteredVacancyList += similarity

 return count(filteredVacancyList)
```

Fig. 3. Filtering algorithm written in pseudocode

The second approach is to use a keyword search including keywords from the contextual domain. Procedure for this approach can be described as follows:

1. Preparation of a list of keywords for the contextual domain.
2. Indexation of unfiltered vacancies. The index includes both vacancy description and its title.
3. Next step is the search for vacancies matching the following condition: the text should contain the keyword and at least one of the contextual keywords.

List of contextual keywords can be prepared once for each category and can be reused. The list itself was formed of the keywords with significantly higher frequency of appearance in the domain compared to the frequency of their occurrence in a common text. An additional manual selection of contextual keywords was performed after the initial automated selection to eliminate the ambiguous words, such as "basic", which is both a commonly used adjective and the name of the programming language.

3 Results

The experiment shows that a filter based on the cosine similarity reduces the percentage of noise in the vacancy list (fig. 4). The values in the chart represent the difference between the raw number of vacancies and the number of vacancies after application of the filtering algorithm. Mean decrease of the noise is about 15.34%. However, in some cases, the filter result shows lower number of vacancies than the actual one. It may affect the representativeness of the skill assessment.

As is evident from fig. 5, the approach based on contextual keywords shows higher accuracy. The amount of noise vacancies decreased by 91.7%. Miscalculation on keywords "MySQL" and "Microsoft Access" might be related to their wide distribution outside software development sector. Therefore, contextual keywords in the software development sector do not always occur in the vacancies containing mentioned keywords, which leads to an underestimation of the number of results.

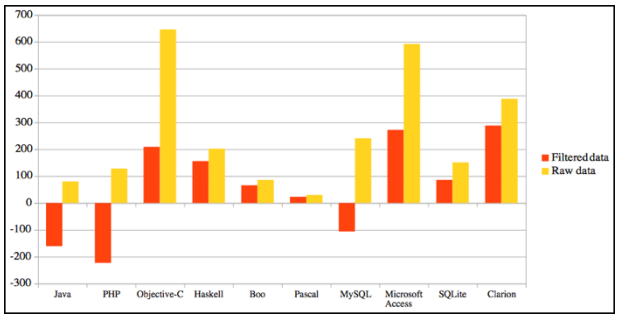

Fig. 4. Results of the algorithm based on the cosine similarity

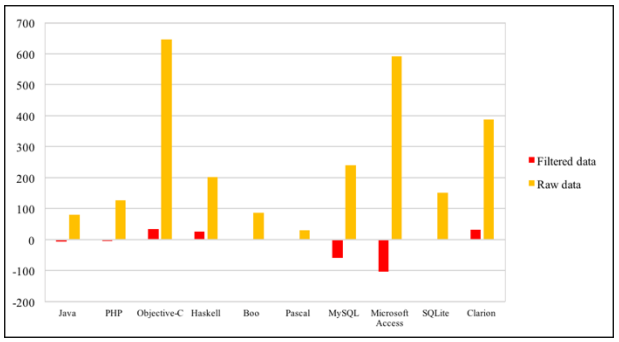

Fig. 5. Results of the algorithm based on the contextual keywords

4 Discussion

Both solutions have their advantages and disadvantages.

The performance of the filter based on the cosine similarity is acceptable to solve problems with a limited set of data, as the complexity of calculating the cosine similarity between all the texts is $O(n^2)$. Due to the fact that the algorithm can underestimate the actual number of vacancies, it is reasonable to carry out further studies in order to improve the accuracy of the algorithm.

Filter based on the contextual keywords has a lower computation complexity $O(n \log(n))$ and shows higher accuracy. But it requires manual preparation of a list of contextual keywords prior to the filtering. As a result, filtering can be performed on a limited number of professional scopes. An algorithm for automatic selection of contextual keywords can be implemented to expand a given set of professional scopes, that would eliminate the manual stage of preparation and will provide an opportunity to apply the method to any area of job search.

5 Conclusion

It is reasonable to integrate both solutions into the predictive learning service and to monitor the sustainability of both approaches to changes in the labor market.

The method based on the contextual keywords provides an ability to carry out studies to identify the most demanded skills in the labor market with high accuracy [9].

References

1. R. T. Fielding, R. N. Taylor, ACM Trans. Internet Technol., **2**, 407-416 (2002)
2. Indeed Publisher Program, https://www.indeed.com/publisher (Accessed: 01/12/2017)
3. HeadHunter API, https://dev.hh.ru/ (Accessed: 01/12/2017)
4. E. Nikulchev, D. Ilin, E. Matishuk, Procedia Computer Science, **103**, 44-51 (2017)
5. T. Joachims, ECML, 137-142 (1998)
6. K. Nigam, A. McCallum, S. Thrun, T. Mitchell, Mach. Learn., **39**, 103-134 (2000)
7. T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Adv. Neural Inf. Process. Syst., 3111-3119 (2013)
8. Q. Le, T. Mikolov, ICML, **32**, 1188-1196 (2014)
9. D. Ilin, D. Strunitsyn et al., ITM Web of Conference, **29**, 02017 (2016)