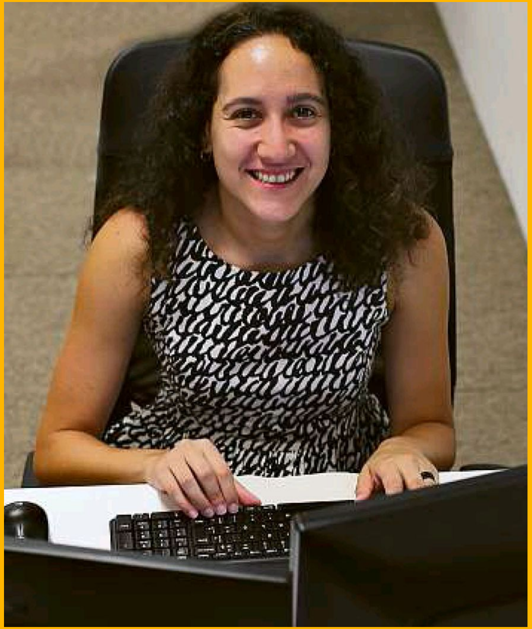
A FRAMEWORK TO SUPPORT TEACHERS IN IDENTIFYING KNOWLEDGE GAPS

EPFL PhD Summit
Advisor: Carlos Eduardo Pedreira
Laura de Oliveira F. Moraes

About me



- PhD Student at UFRJ, Brazil
    - *Research field:* Artificial Intelligence, Education.
    - Machine Teaching: This project aims to use elements on how machines learn to improve how students learn.

- Software Engineer at CERN for 4 years
- 2018 Fellow at Data Science for Social Good
- Data Science startup co-founder (TWIST Systems)

Motivation

- Understand gaps in knowledge **continuously**
  - Students are evaluated only in specific times (pre-exam anxiety)
  - Improve teaching methods
  - Personalized education

- Not domain specific

- Python language
  - Fastest-growing major programming language

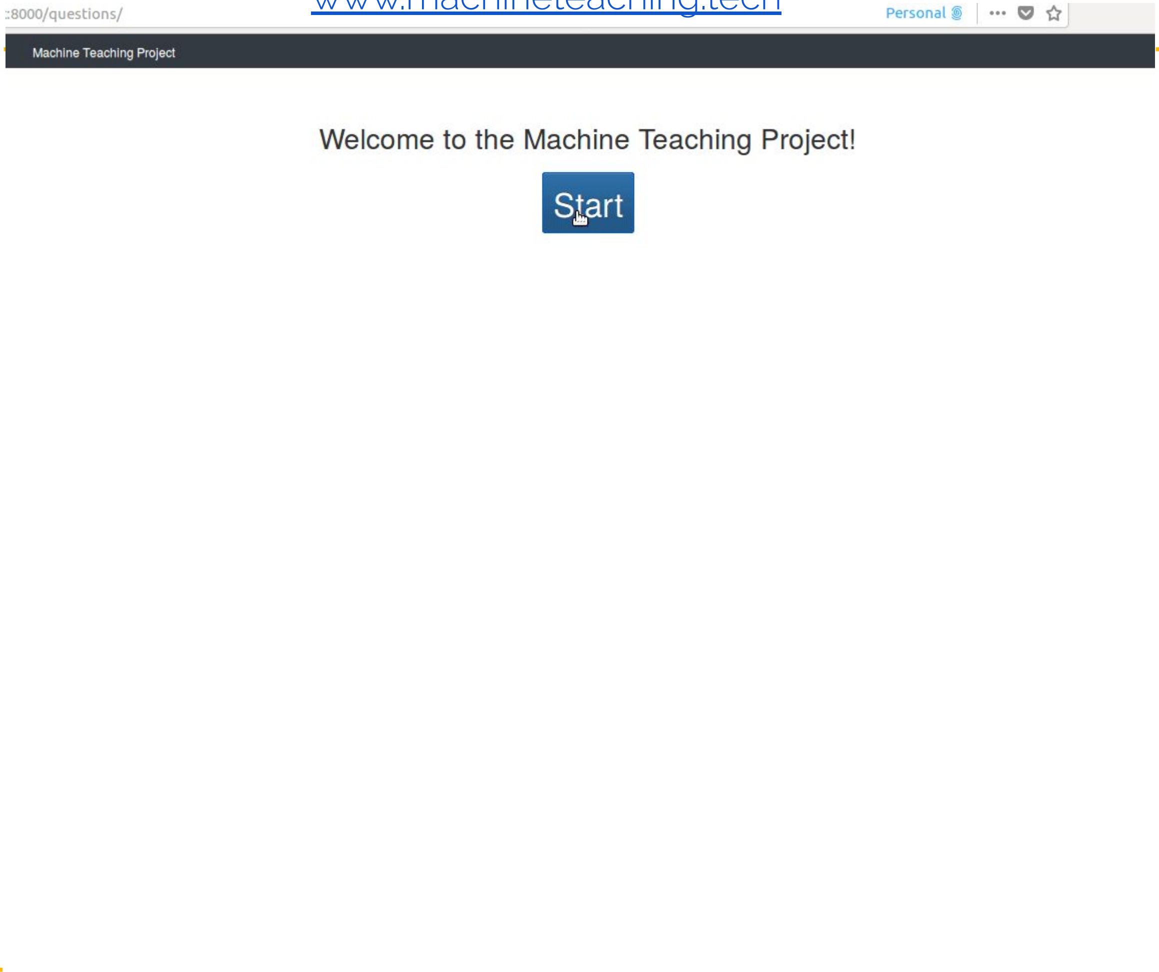
Goal

Assess students' knowledge continuously, avoiding pre-exam anxiety and late diagnosis.

★ Input: questions, concepts and a student

★ Output: student knowledge inference

www.machineteaching.tech



Contributions so far...

✓ An open-source anonymized database:

a. **435 college freshman** with at least one submission

b. **854** different **problems** (90 revised)

c. **21,884** different **submissions** (in 5,607 problems)

d. Running this semester in **4 classes**

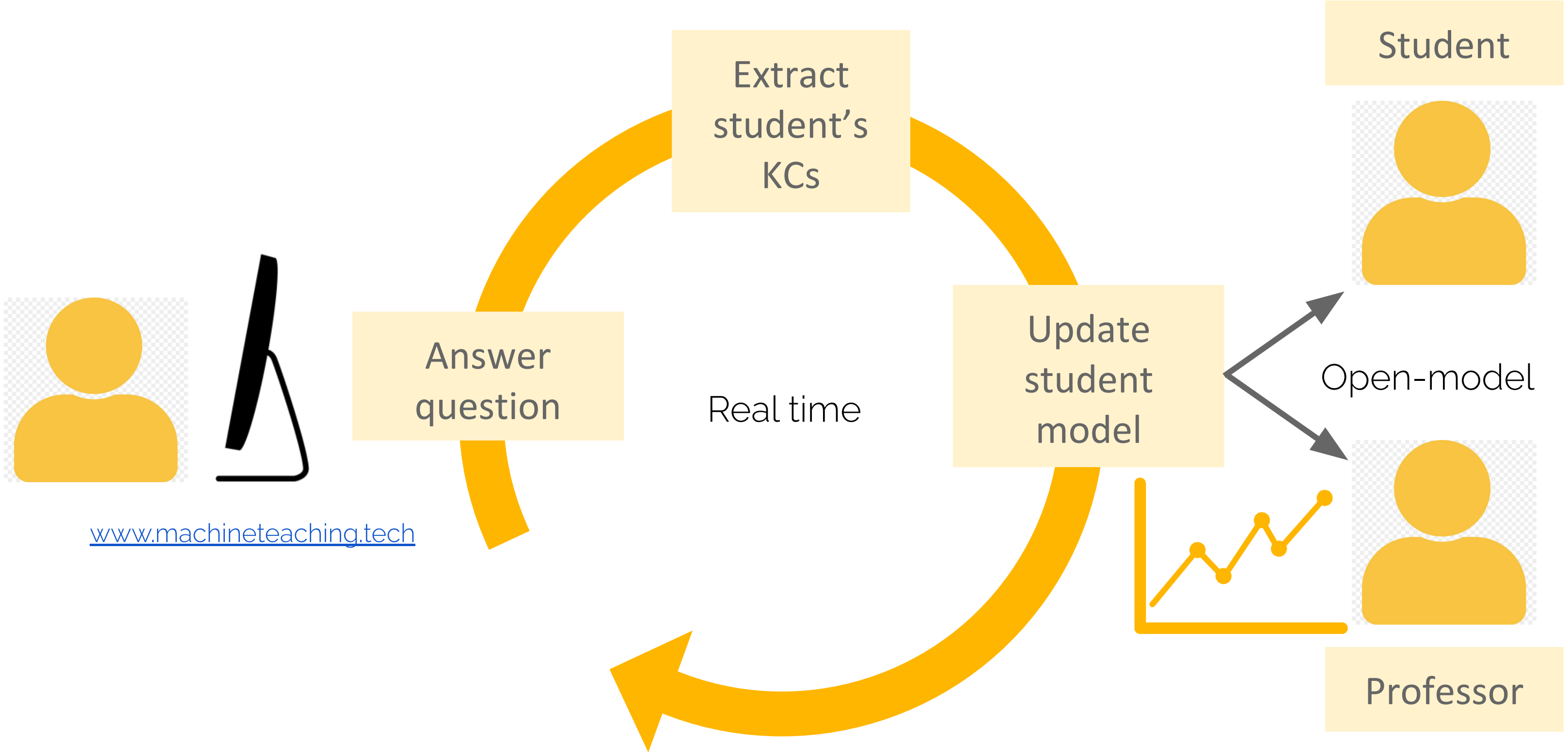
✓ Unsupervised knowledge component (KC) discovery

✓ Personalized student model.

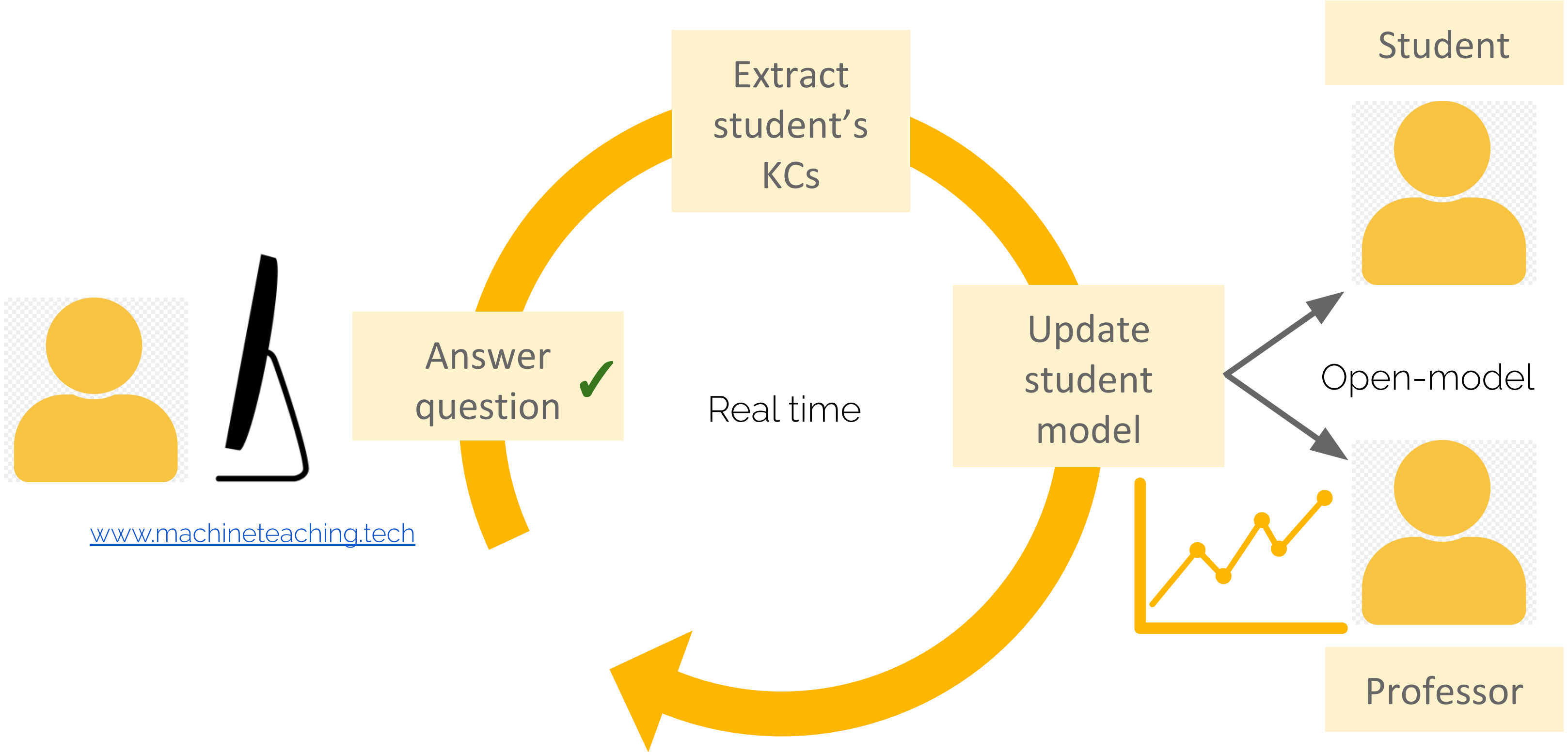
✓ Real user studies.

✓ Integrated framework

How does it work?

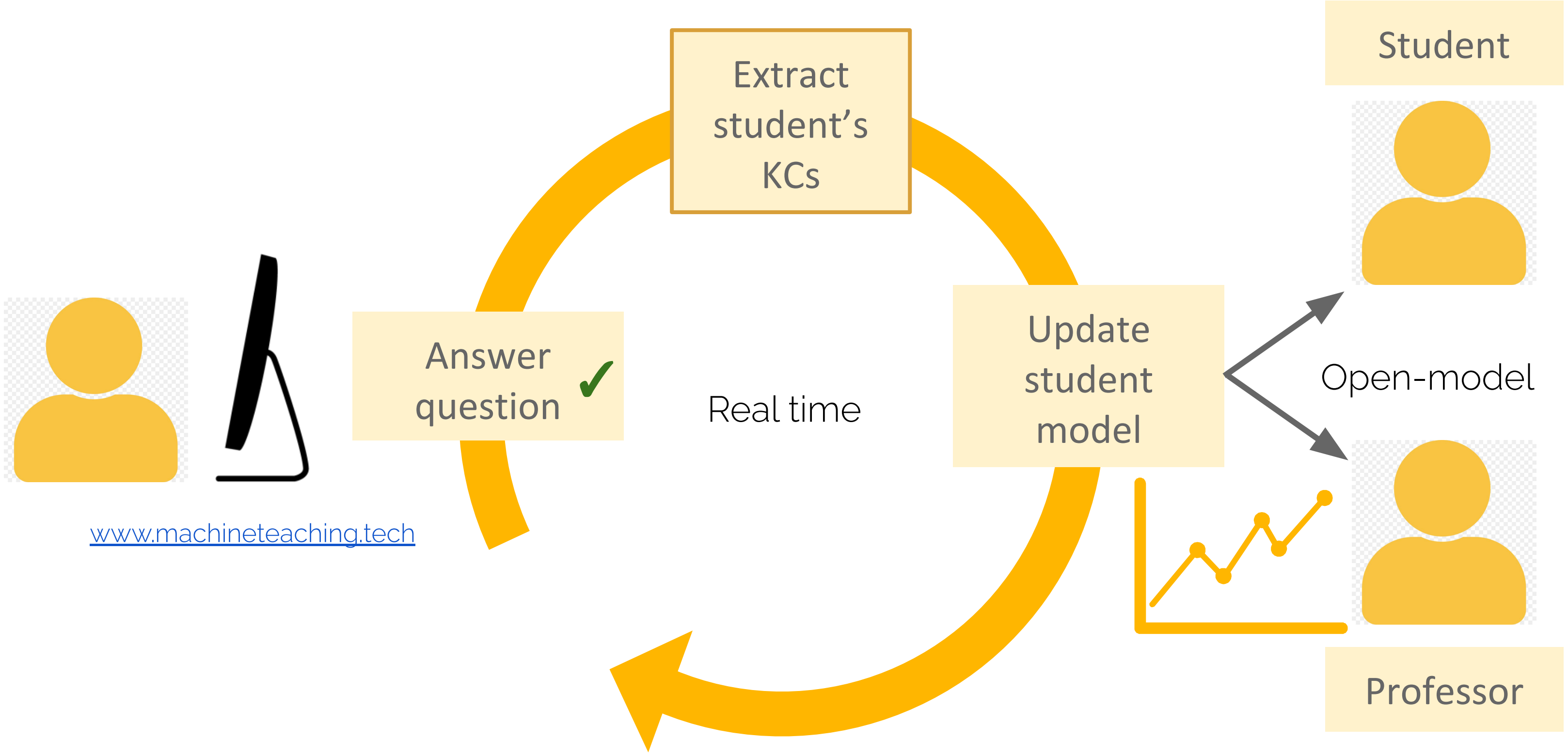


How does it work?



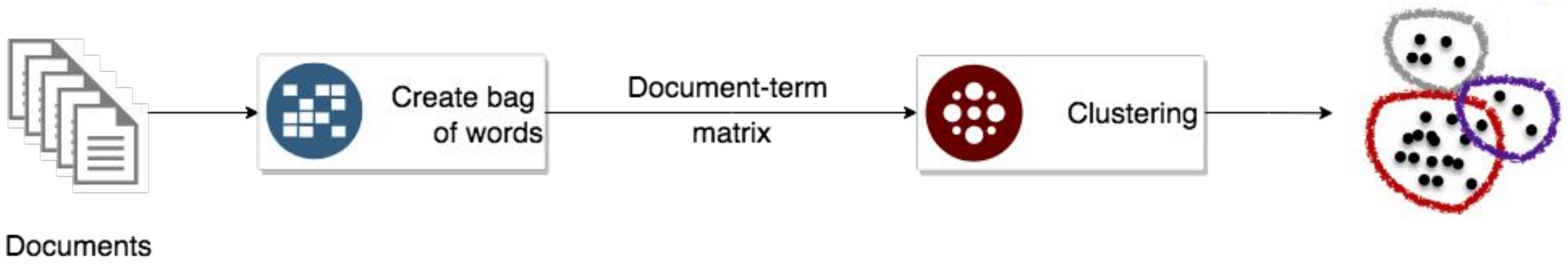
www.machineteaching.tech

How does it work?

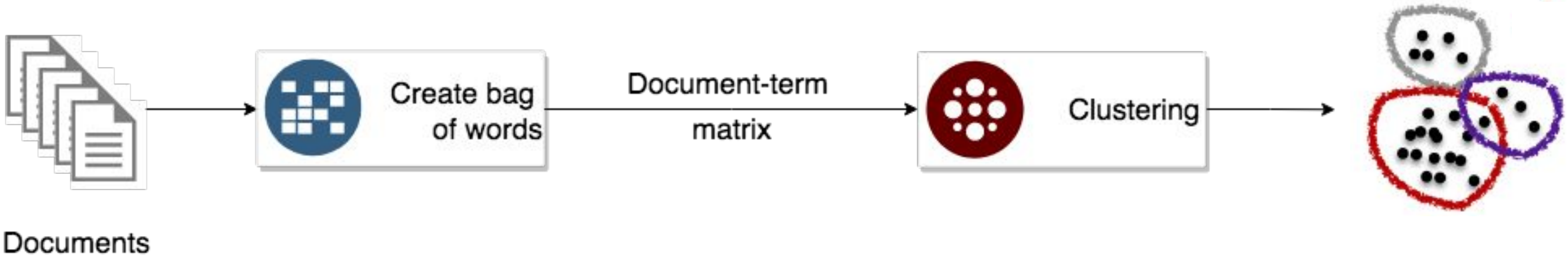


www.machineteaching.tech

Methodology



Methodology



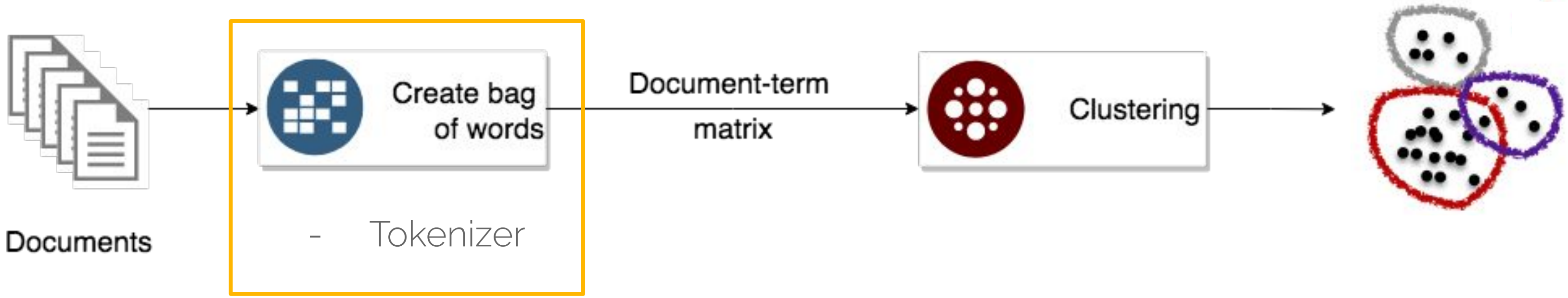
Document-term matrix

terms/tokens/words

documents
code snippets
answers

	for	if	append
1	2	1	1
2	0	1	1
3	1	2	0

Methodology



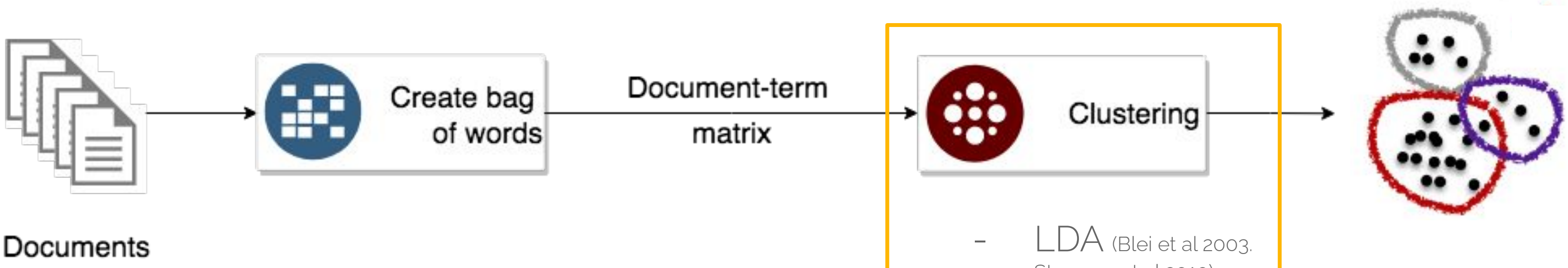
Document-term matrix

terms/tokens/words

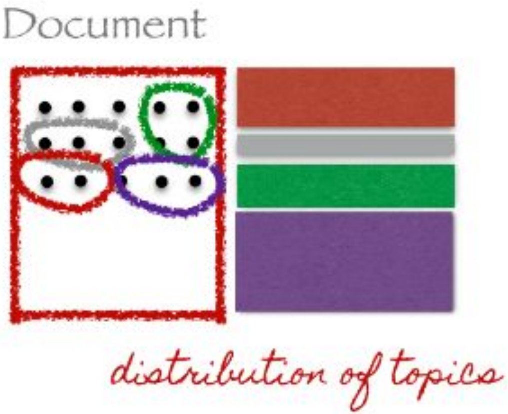
documents
code snippets
answers

	for	if	append
1	2	1	1
2	0	1	1
3	1	2	0

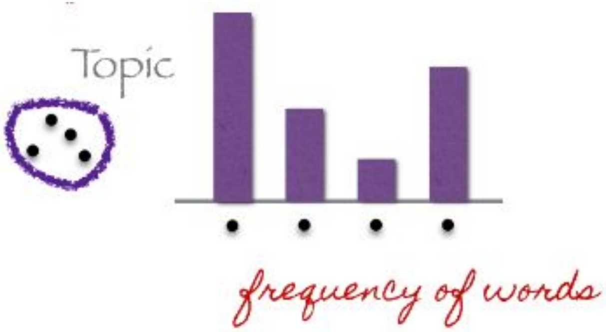
Methodology



- LDA (Blei et al 2003, Steyvers et al 2010)
- NMF (Lee et al 1999, Cichocki et al 2009)



- words/tokens



Given all these possibilities, how to choose the best set of hyper-parameters, the clustering method and the number of clusters?

Evaluation - Topic Coherence

(Mimno et al 2011, Röder et al 2015)

- **Human interpretability** metric

- Ratio between **co-occurence of top-N** words and their **total occurence** within the topic.

- Same words appear in the same documents of the topic.

$$C_{UMass}(t_k, V) = \sum_{m=1}^{N} \sum_{j=1}^{N} log \frac{Q(v_j, v_m, t_k) + \epsilon}{Q(v_j, t_k)}$$

Methodology

Grid search (Bergstra et al 2012)

10 Minimum Document Frequency Values X 2 Binary Appearances Options X 3 Vectorizers X 2 Clustering Methods X 14 Number of Clusters/Topics = 1680 RUNS

Dataset

```python
def square(number):
    square_dict=dict()
    for i in range(1,number+1):
        square_dict[i]=i*i
    return square_dict
```

```python
def fatorial(number):
    total = 1
    for i in range(number, 1, -1):
        total = total * i
    return total
```

54 revised Python snippets

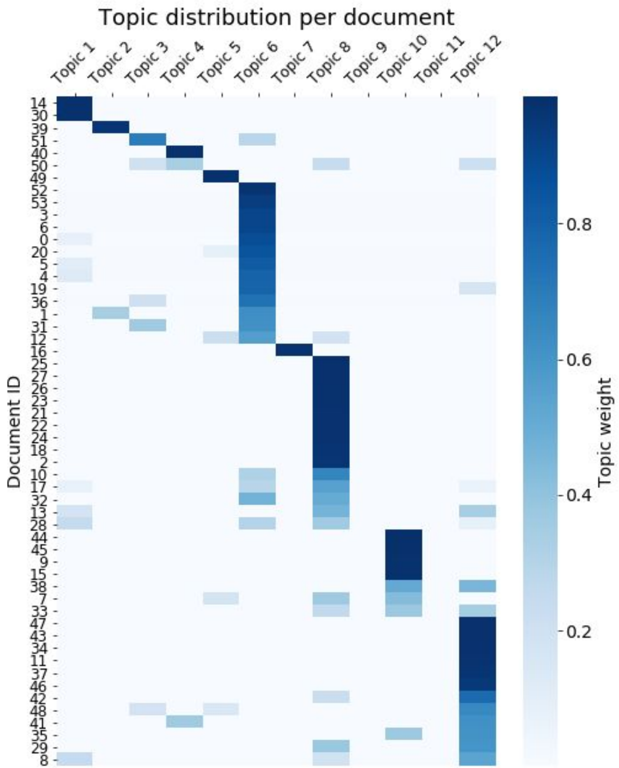
to be used as **documents**

(now, there are 90)

Experiment Id	Min DF	Binary	Vectorizer	Method	Best-k
26	**0.05**	**True**	**Count**	**LDA**	**12**

Document distribution per topic:

- Topic 1: 2
- Topic 2: 1
- Topic 3: 1
- Topic 4: 2
- Topic 5: 1
- **Topic 6: 13**
- Topic 7: 1
- **Topic 8: 14**
- Topic 9: 0
- **Topic 10: 7**
- Topic 11: 0
- **Topic 12: 12**



Turning Topics into Concepts

String manipulation

7. Data type: string
14. Function
8. Data type: array or list

Conditional structure

6. Logic
11. Conditional
14. Function

Math and string loops

5. Math
7. Data type: string
12. Loop

Math functions

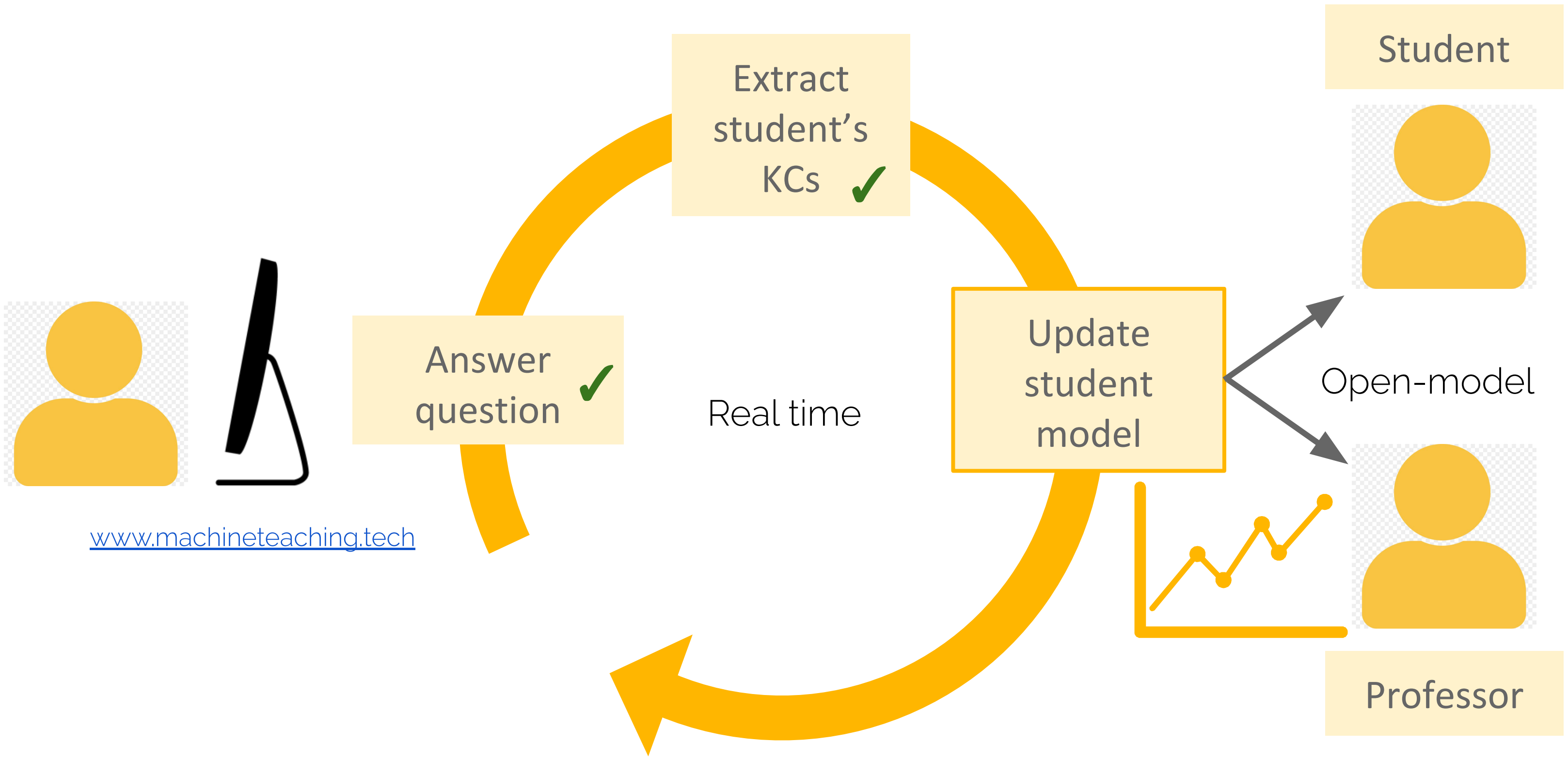
1. Syntax
3. Data type: number
5. Math
14. Function

List loops

8. Data type: list or array
11. Conditional
12. Loop

14 professors evaluated the results in 3 tasks

Review



Extract student's KCs ✓
Answer question ✓
Real time
Update student model
Student
Open-model
Professor
www.machineteaching.tech

Student models

- Bayesian Knowledge Tracing (Corbett et. al. 1995)
  - Been widely used
  - Several enhancement proposals
- Performance Factor Analysis (Pavlik et al 2009)
- Tensor factorization (Sahebi et al 2016)

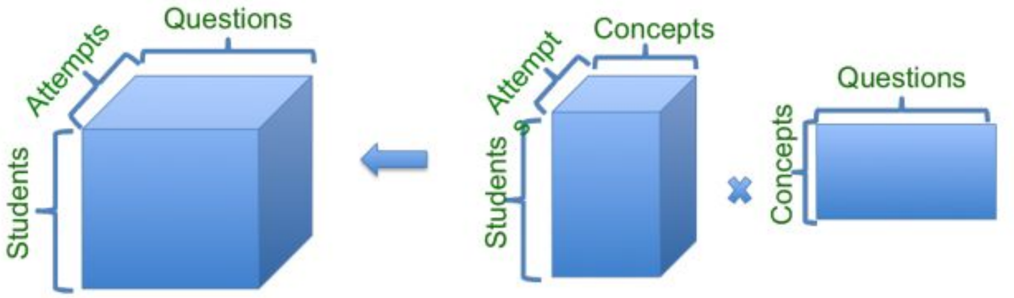


Figure reproduced from Sahebi et al 2016

Student Performance Kit

- Python library like Scikit-Learn
  - 3 models: **BKT**, PFA, TF
  - Scores: **LL, AIC, BIC, RMSE, Acc,** AUC

spkit 0.0.1

```
pip install -i https://test.pypi.org/simple/ spkit==0.0.1
```

Conclusions

- **Engage** a team of students, professors and teaching assistants.
- Novel approach to **code clustering** in the EDM context.
- Good clustering schemes suited for **human interpretability**.
- The majority of the methodology can be **generalized to other domains.**
- Open and reproducible (still needs a bit of organization!):
  - https://github.com/laura-moraes/machine-teaching
  - https://github.com/lauramoraes/StudentPerformanceKit

Next Questions

- How much domain knowledge and specialization is needed to transpose it to another domain, such as History or Social Studies?
- Can we recommend personalized content in order to improve learning?

Thanks!

Any questions?

Students complain a little about the site, but it is because they are not very aware of how useful it is to themselves. For me the level of the practical classes has greatly improved.

Hugo Nobrega, professor

3

Additional material

Related Work

- Autograders

- **Question classification**
  - Specialists
  - Supervised Learning
  - Unsupervised Learning
  - ASTs
  - **Matrix factorization**

CS1 Concepts

4 References were used to define the concepts usually given in a CS1 course:

- ACM Computer Science Curricula 2013
- Sheard et al 2011
- Petersen et al 2011
- Cherenkova et al 2014

LIST OF FINAL CONCEPTS:

1. Syntax
2. Assignment
3. Data type: number
4. Data type: boolean
5. Math
6. Logic
7. Data type: string
8. Data type: list or array
9. Data type: tuple
10. Data type: dict
11. Conditional
12. Loop
13. Nested loop
14. Function
15. Recursion

Create bag of words

Tokenizer — Minimum Document Frequency — Binary Appearance — Vectorizer

Separate text in:

- **Keywords**
- **Operators**
- **Data types**
- **Indents/Dedents**

Choose a value in a range from 0.05 to 0.5 (percentage of documents)

How to count word appearance:

1. Just once per document **OR**
2. Each time it appears in the document

How to count frequency of words:

1. Regular Count **OR**
2. Tf-Idf weights (Salton et al 1986, Zhang et al 2011) **OR**
3. NCut weights (Yan et al 2012)

Clustering *Methods for Topic Modeling*

Latent Dirichlet Allocation (**LDA**)

(Blei et al 2003, Steyvers et al 2010)

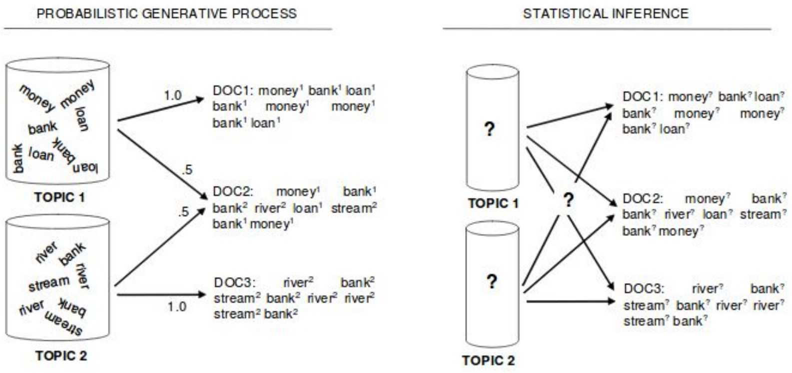


Figure reproduced from Steyvers et al 2010

Non-negative Matrix Factorization (**NMF**)

(Lee et al 1999, Cichocki et al 2009)

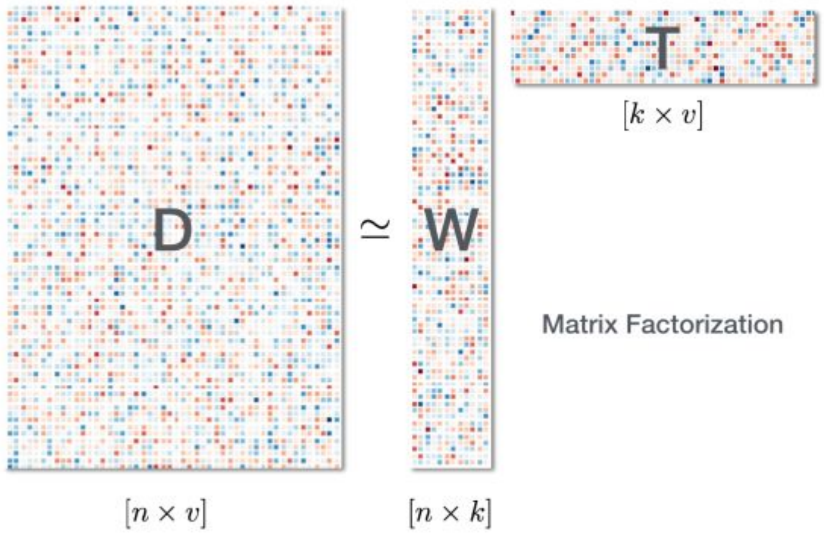


Figure reproduced from http://tech.opentable.com/2015/01/12/finding-key-themes-from-free-text-reviews/

Experiments

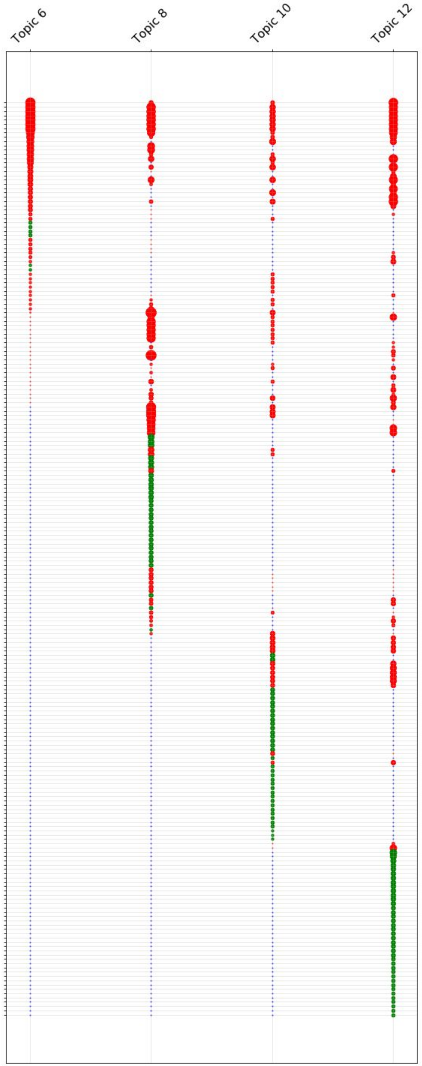
Objective: group code snippets used to solve CS1 exercises according to the concepts used to solve the problem.

Hypothesis: the words used to code a CS1 exercise are an indicative of the concepts needed to solve them.

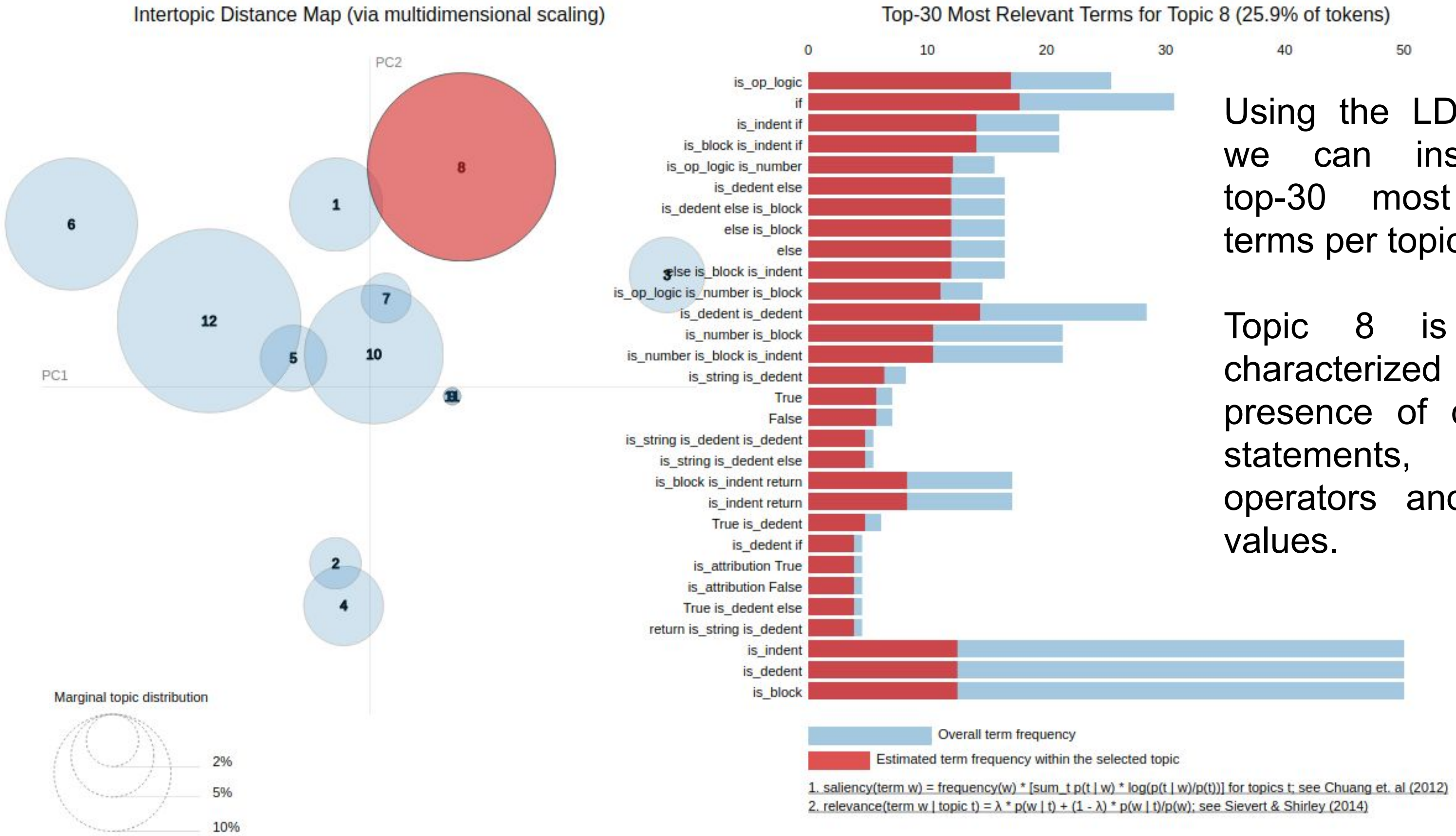
Methodology: cluster these code snippets using as bag of words the words used in the code and the presented methodology.

Taking a deeper look

The main topics 6, 8, 10 and 12 correspond to 85% of the documents and 77.4% of the terms.



Topic 8



Using the LDAVis tool, we can inspect the top-30 most relevant terms per topic.

Topic 8 is strongly characterized by the presence of conditional statements, logical operators and boolean values.

Topic 8: code snippets

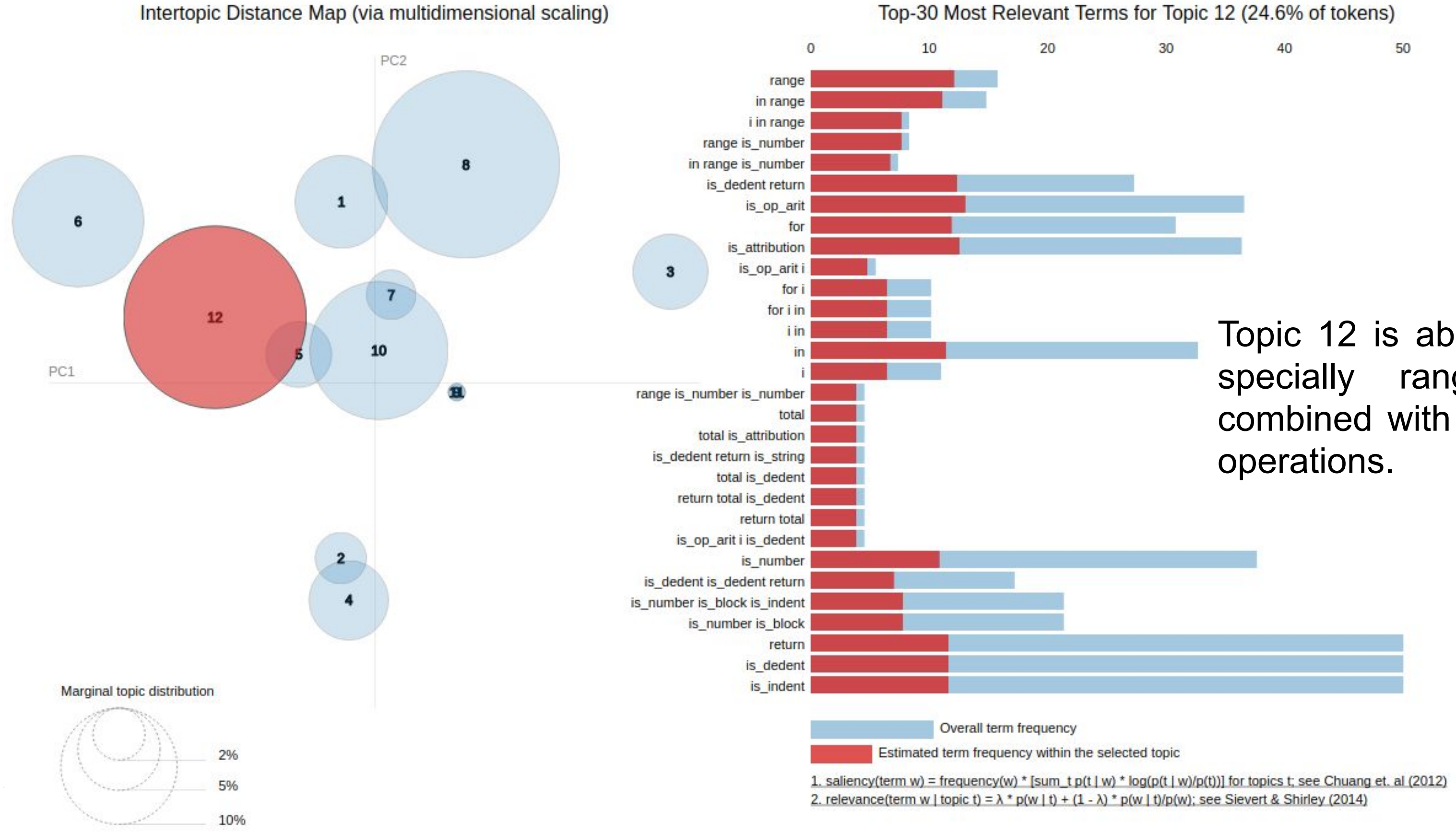
```python
def is_prime(number):
    '''Returns True for prime numbers, False otherwise'''
    #Edge Cases
    if number == 1:
        prime = False
    elif number == 2:
        prime = True
    #All other primes
    else:
        prime = True
        for check_number in range(2, int(number/2)+1):
            if number % check_number == 0:
                prime = False
                break
    return prime
```

```python
def max_of_three(a,b,c):
    max_number = 0
    if a > b:
        if a > c:
            max_number = a
        else:
            max_number = c
    else:
        if b > c:
            max_number = b
        else:
            max_number = c
    return max_number
```

```python
def light(switchA, switchB):
    if switchA == 1 and switchB == 1:
        return True
    else:
        return False
```

```python
def days_in_month(month, year):
    if month == 2:
        if (year % 400) == 0:
            return 29
        elif (year % 100) == 0:
            return 28
        elif (year % 4) == 0:
            return 29
        else:
            return 28
    elif month in (4,6,9,11):
        return 30
    elif month in (1,3,5,7,8,10, 12):
        return 31
```

Topic 12



Topic 12 is about loops, specially range loops combined with arithmetic operations.

Topic 12: code snippets

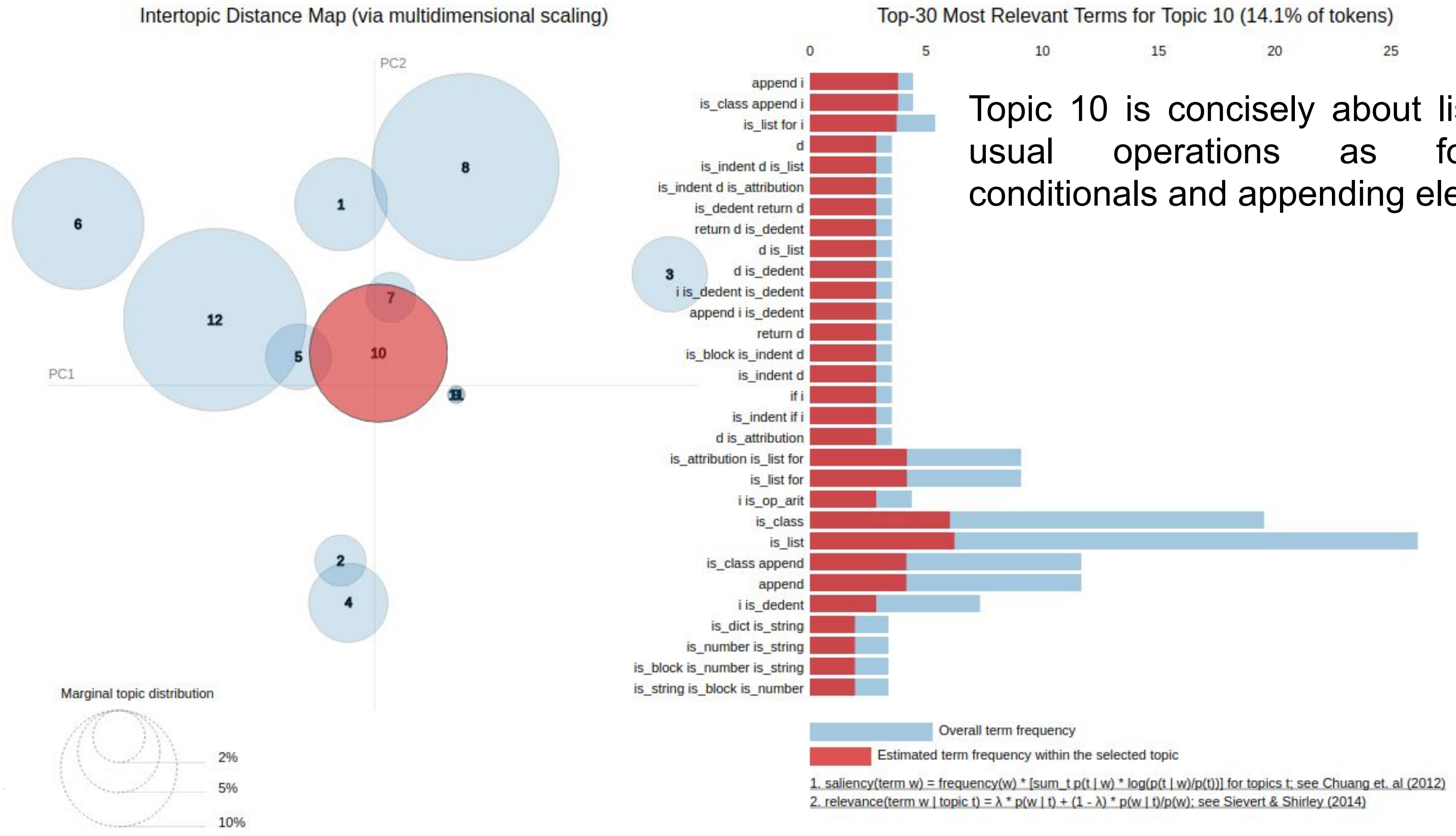
```
def digit_sum(digit):
    total = 0
    for i in range(1,5):
        number = "%s" % digit
        number = int(number * i)
        total = total + number
    return total
```

```
def all_even():
    values = []
    for a in range(0,9,2):
        for b in range(0,9,2):
            for c in range(0,9,2):
                values.append('2'+str(a)+str(b)+str(c))
    return ", ".join(values)
```

```
def fatorial(number):
    total = 1
    for i in range(number, 1, -1):
        total = total * i
    return total
```

```
def binby5(number_csv):
    value = []
    items=[x for x in number_csv.split(',')]
    for p in items:
        intp = int(p, 2)
        intp = int(p[0])*8 + int(p[1])*4 + int(p[2])*2 + int(p[3])
        if not intp%5:
            value.append(p)
    return ", ".join(value)
```

Topic 10



Topic 10 is concisely about lists and its usual operations as for loops, conditionals and appending elements.

Topic 10: code snippets

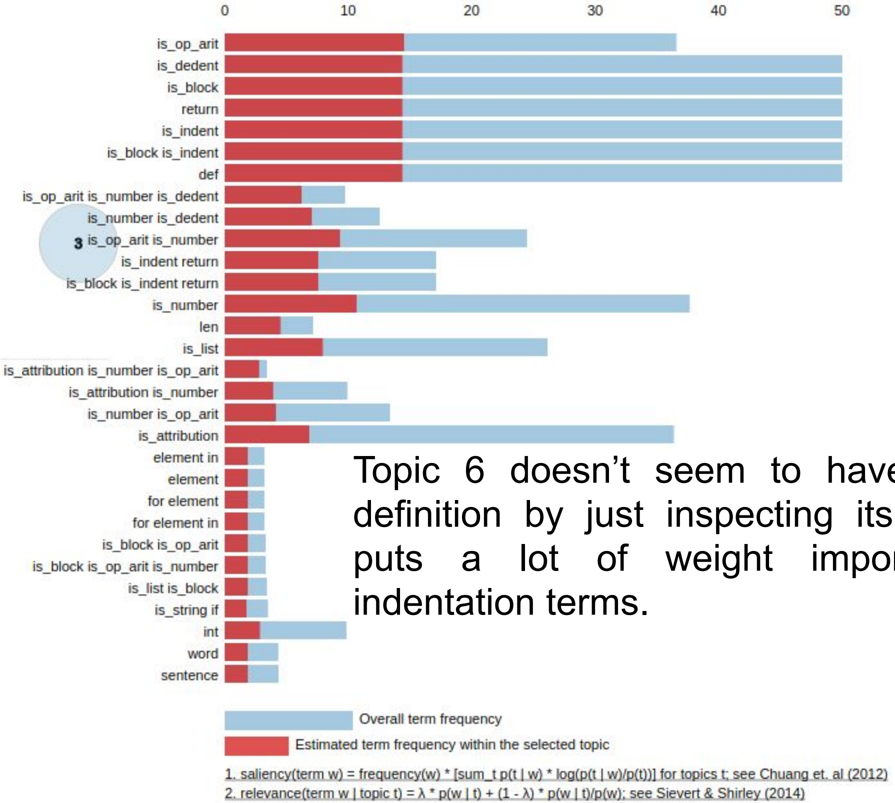
```
def common(list1, list2):
    common_list = []
    for i in list1:
        if i in list2 and i not in common_list:
            common_list.append(i)
    common_list.sort()
    return common_list
```

```
def dedupe(dup_list):
    nodup_list = []
    for i in dup_list:
        if i not in nodup_list:
            nodup_list.append(i)
    return nodup_list
```

```
def count(sentence):
    d={"digits":0, "letters":0}
    for char in sentence:
        if char.isdigit():
            d["digits"]+=1
        elif char.isalpha():
            d["letters"]+=1
        else:
            pass
    return d
```

```
def divisible():
    l=[]
    for i in range(2000, 3201):
        if (i%7==0) and (i%5!=0):
            l.append(i)
    return l
```

Topic 6



Topic 6 doesn't seem to have a clear definition by just inspecting its terms. It puts a lot of weight importance in indentation terms.

Topic 6: code snippets

```
import math
def formula(D):
    C = 50
    H = 30
    Q = round(math.sqrt(2*C*D/float(H)))
    return Q
```

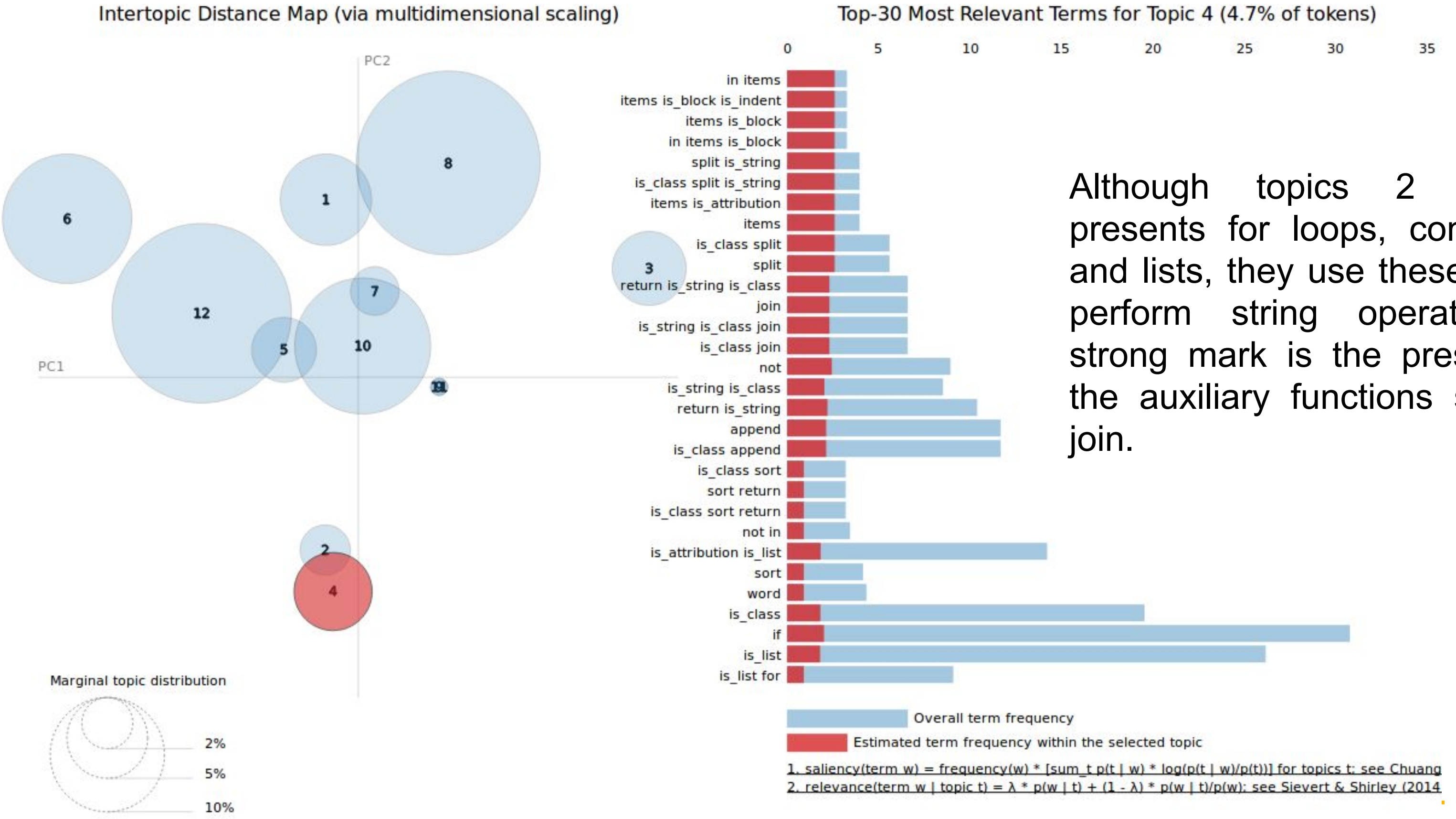
By analyzing the code snippets, topic 6 comprises codes with one indentation structure (simple coding structures), sometimes without even the need to assign variables to solve the exercise.

```
def sum_str(s1,s2):
    return int(s1)+int(s2)
```

```
def square(num):
    return num ** 2
```

```
def euro_conversion(amount, exchange_rate):
    euro = int(amount//exchange_rate)

    euro50s = int(euro // 50)
    remainingEuros = euro % 50

    euro20s = int(remainingEuros // 20)
    remainingEuros = remainingEuros % 20

    euro10s = int(remainingEuros // 10)
    remainingEuros = remainingEuros % 10

    euro5s = int(remainingEuros // 5)
    remainingEuros = remainingEuros % 5

    return(euro, euro50s, euro20s, euro10s, euro5s, remainingEuros)
```

Topics 2 and 4



Although topics 2 and 4, presents for loops, conditionals and lists, they use these tools to perform string operations. A strong mark is the presence of the auxiliary functions split and join.

Topics 2 and 4: code snippets

```
def check_password(passwords):
    import re
    value = []
    items=[x for x in passwords.split(',')]
    for p in items:
        if len(p)<6 or len(p)>12:
            continue
        else:
            pass
        if not re.search("[a-z]",p):
            continue
        elif not re.search("[0-9]",p):
            continue
        elif not re.search("[A-Z]",p):
            continue
        elif not re.search("[$#@]",p):
            continue
        elif re.search("\s",p):
            continue
        else:
            pass
        value.append(p)
    return ",".join(value)
```

```
def sort_dedupe(words):
    items = words.split(' ')
    items_dedupe = []
    for word in items:
        if word not in items_dedupe:
            items_dedupe.append(word)
    items_dedupe.sort()
    return " ".join(items_dedupe)
```

```
def sort_csv(csv):
    items = csv.split(', ')
    items.sort()
    return ", ".join(items)
```

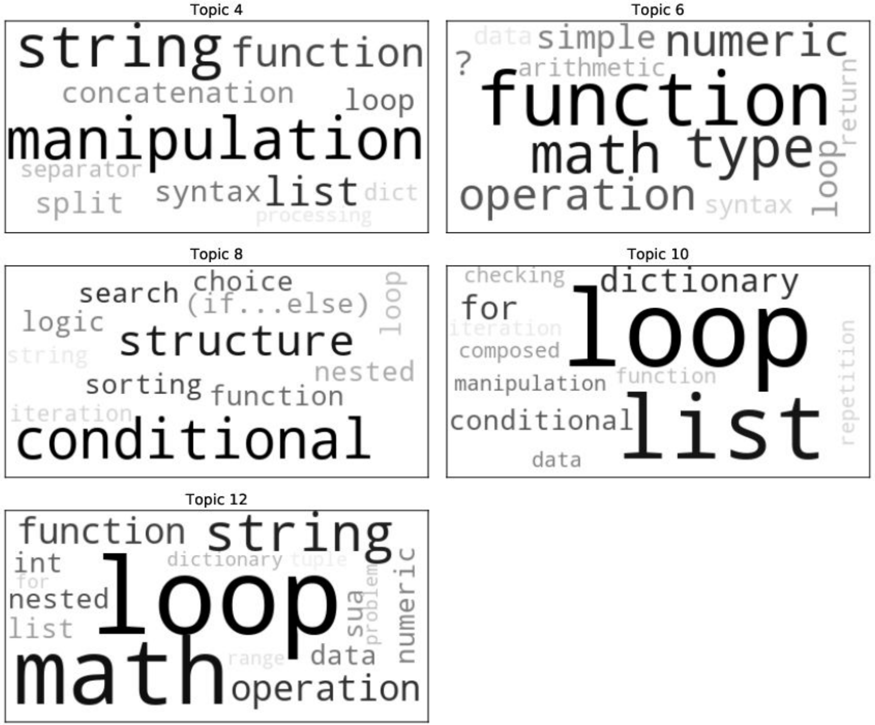
Turning Topics into Concepts

- **Theme identification**: the professors were shown four codes from the same topic. They should **label each topic** with a simple description.

- **Concept identification**: each professor should **associate up to three concepts** (from the 15 available) to each presented code.

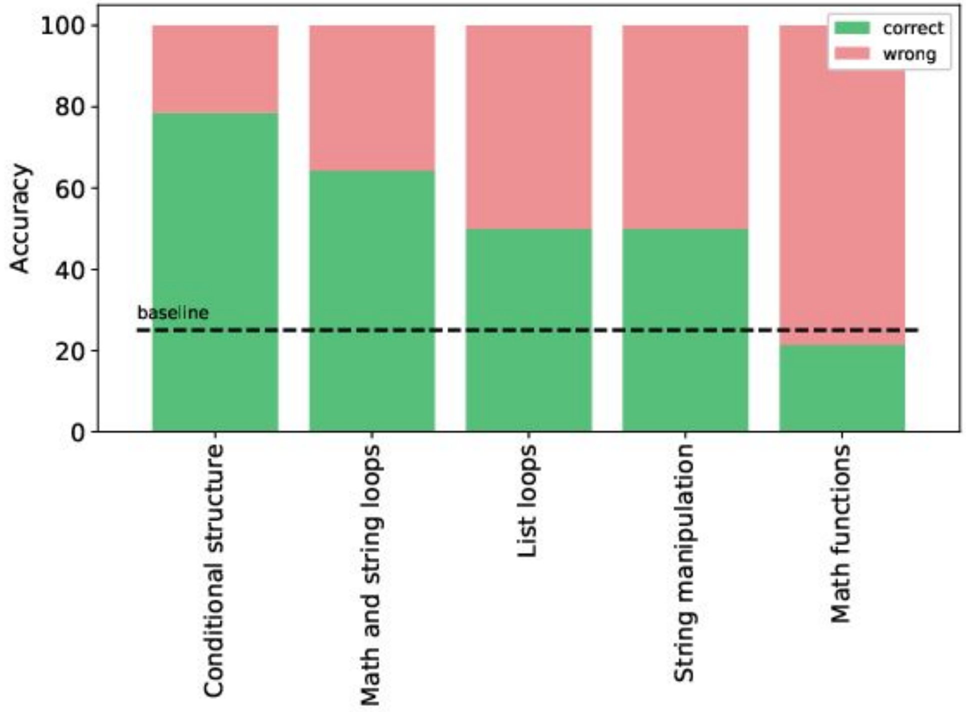
- **Intruder identification:** the professors should **identify the intruder document given a topic.**

Theme identification



- Topic 4: String manipulation
- Topic 6: Math functions
- Topic 8: Conditional structure
- Topic 10: List loops
- Topic 12: Math and string loops

Intruder identification



References

ACM Computer Science Curricula 2013

Bergstra, J., & Yoshua Bengio, U. (2012). Random Search for HyperParameter Optimization. Journal of Machine Learning Research. https://doi.org/10.1162/153244303322533223

Blei. David M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research. https://doi.org/10.1162/jmlr.2003.3.4-5.993

Cherenkova, Y., Zingaro, D., & Petersen, A. (2014). Identifying challenging CS1 concepts in a large problem dataset. In Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14. https://doi.org/10.1145/2538862.2538966

Cichocki, A., & Phan, A. H. (2009). Fast local algorithms for large scale nonnegative matrix and tensor factorizations. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. https://doi.org/10.1587/transfun.E92.A.708

Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction. https://doi.org/10.1007/BF0109982

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. Nature. https://doi.org/10.1038/44565

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., & McCallum, A. (2011). Optimizing Semantic Coherence in Topic Models. In Proc. of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11. https://doi.org/10.1037/1082-989X.12.1.105

References

Pavlik, P. I., Cen, H., & Koedinger, K. R. (2009). Performance factors analysis - A new alternative to knowledge tracing. In Frontiers in Artificial Intelligence and Applications. https://doi.org/10.3233/978-1-60750-028-5-531

Petersen, A., Craig, M., & Zingaro, D. (2011). Reviewing CS1 exam question content. In Proceedings of the 42nd ACM technical symposium on Computer science education - SIGCSE '11. https://doi.org/10.1145/1953163.1953340

Röder, M., Both, A., & Hinneburg, A. (2015). Exploring the Space of Topic Coherence Measures. In Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15. https://doi.org/10.1145/2684822.2685324

Sahebi, S., Lin, Y.-R., & Brusilovsky, P. (2016). Tensor Factorization for Student Modeling and Performance Prediction in Unstructured Domain. In Educational Data Mining, proceedings of the 9th International Conference on. http://www.educationaldatamining.org/EDM2016/proceedings/paper_150.pdf

Salton, G., & McGill, M. J. (1986). Introduction to modern information retrieval. Introduction to Information Retrieval (p. 400). McGraw-Hill, Inc. Retrieved from http://portal.acm.org/citation.cfm?id=576628

Sheard, J., Carbone, A., Clear, T., Raadt, M. De, Souza, D. D., Harland, J., … Warburton, G. (2011). Exploring Programming Assessment Instruments : A Classification Scheme for Examination Questions. In Seventh International Workshop on Computing Education Research. https://doi.org/10.1145/2016911.2016920

References

Steyvers, M., & Griffiths, T. (2010). Probalistic Topic Models. In Latent Semantic Analysis: A Road To Meaning. https://doi.org/10.1016/s0364-0213(01)00040-4

Zhang, W., Yoshida, T., & Tang, X. (2011). A comparative study of TF*IDF, LSI and multi-words for text classification. Expert Systems with Applications. https://doi.org/10.1016/j.eswa.2010.08.066

Yan, X., Guo, J., Liu, S., Cheng, X., & Wang, Y. (2012). Clustering short text using Ncut-weighted non-negative matrix factorization. In Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12 (p. 2259). https://doi.org/10.1145/2396761.2398615