Mining Massive Datasets
Project Progress

Diversity of Wikipedia article references

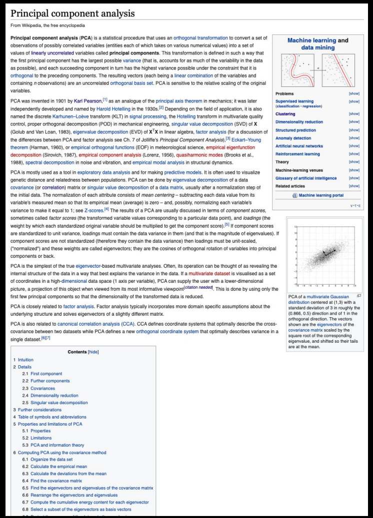
Yaroslava Lochman
Philipp Kofman
Sasha Chepurnoi
Vadym Korshunov

July 2019

Outline

- Goals / Problem statement
- Motivation
- Work pipeline
  - Overview
  - Data processing + Feature Engineering
  - Modeling
  - Evaluation
- Further work

Goals / Problem Statement



Principal component analysis

From Wikipedia, the free encyclopedia

Principal component analysis (**PCA**) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called **principal components**. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors (each being a linear combination of the variables and containing *n* observations) are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

PCA was invented in 1901 by Karl Pearson,[1] as an analogue of the principal axis theorem in mechanics; it was later independently developed and named by Harold Hotelling in the 1930s.[2] Depending on the field of application, it is also named the discrete Karhunen–Loève transform (KLT) in signal processing, the Hotelling transform in multivariate quality control, proper orthogonal decomposition (POD) in mechanical engineering, singular value decomposition (SVD) of $\mathbf{X}$ (Golub and Van Loan, 1983), eigenvalue decomposition (EVD) of $\mathbf{X}^T\mathbf{X}$ in linear algebra, factor analysis (for a discussion of the differences between PCA and factor analysis see Ch. 7 of Jolliffe's *Principal Component Analysis*),[3] Eckart–Young theorem (Harman, 1960), or empirical orthogonal functions (EOF) in meteorological science, empirical eigenfunction decomposition (Sirovich, 1987), empirical component analysis (Lorenz, 1956), quasiharmonic modes (Brooks et al., 1988), spectral decomposition in noise and vibration, and empirical modal analysis in structural dynamics.

PCA is mostly used as a tool in exploratory data analysis and for making predictive models. It is often used to visualize genetic distance and relatedness between populations. PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix, usually after a normalization step of the initial data. The normalization of each attribute consists of *mean centering* – subtracting each data value from its variable's measured mean so that its empirical mean (average) is zero – and, possibly, normalizing each variable's variance to make it equal to 1; see Z-scores.[4] The results of a PCA are usually discussed in terms of *component scores*, sometimes called *factor scores* (the transformed variable values corresponding to a particular data point), and *loadings* (the weight by which each standardized original variable should be multiplied to get the component score).[5] If component scores are standardized to unit variance, loadings must contain the data variance in them (and that is the magnitude of eigenvalues). If component scores are not standardized (therefore they contain the data variance) then loadings must be unit-scaled, ("normalized") and these weights are called eigenvectors; they are the cosines of orthogonal rotation of variables into principal components or back.

PCA is the simplest of the true eigenvector-based multivariate analyses. Often, its operation can be thought of as revealing the internal structure of the data in a way that best explains the variance in the data. If a multivariate dataset is visualised as a set of coordinates in a high-dimensional data space (1 axis per variable), PCA can supply the user with a lower-dimensional picture, a projection of this object when viewed from its most informative viewpoint[citation needed]. This is done by using only the first few principal components so that the dimensionality of the transformed data is reduced.

PCA is closely related to factor analysis. Factor analysis typically incorporates more domain specific assumptions about the underlying structure and solves eigenvectors of a slightly different matrix.

PCA is also related to canonical correlation analysis (CCA). CCA defines coordinate systems that optimally describe the cross-covariance between two datasets while PCA defines a new orthogonal coordinate system that optimally describes variance in a single dataset.[6][7]

Machine learning and data mining

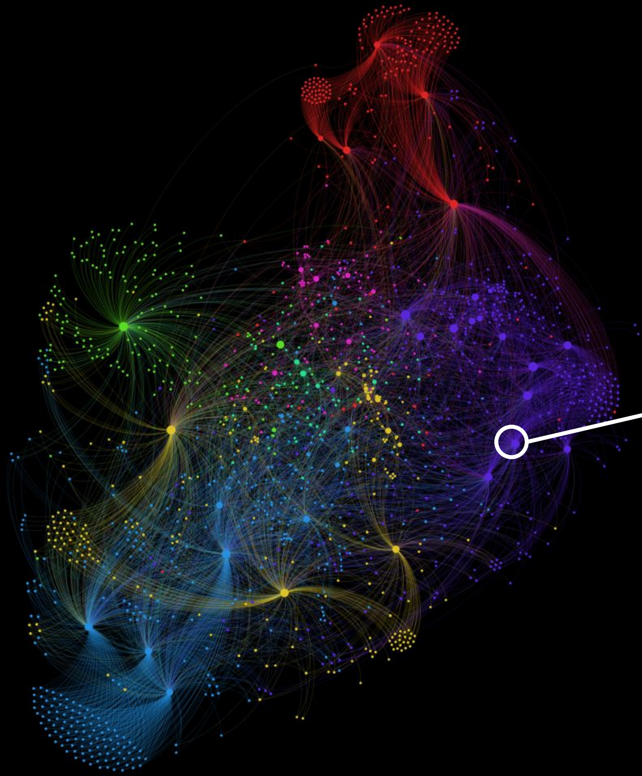
Problems	[show]
Supervised learning (classification · regression)	[show]
Clustering	[show]
Dimensionality reduction	[show]
Structured prediction	[show]
Anomaly detection	[show]
Artificial neural networks	[show]
Reinforcement learning	[show]
Theory	[show]
Machine-learning venues	[show]
Glossary of artificial intelligence	[show]
Related articles	[show]

Machine learning portal

V · T · E

PCA of a multivariate Gaussian distribution centered at (1,3) with a standard deviation of 3 in roughly the (0.866, 0.5) direction and of 1 in the orthogonal direction. The vectors shown are the eigenvectors of the covariance matrix scaled by the square root of the corresponding eigenvalue, and shifted so their tails are at the mean.

Contents [hide]

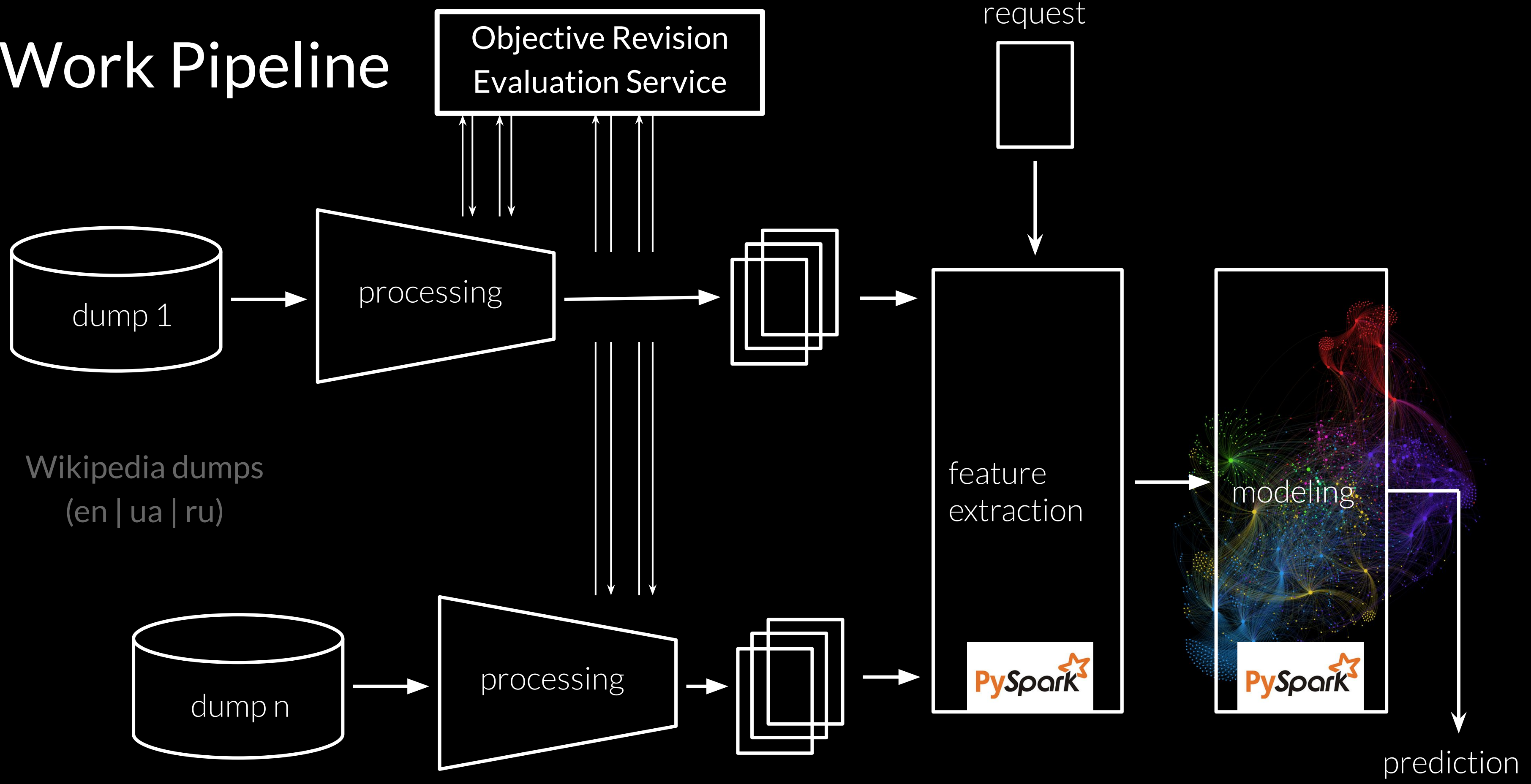


- the page is well cited and argumented
- the references distribution:
  55%: scientific papers
  40%: books
  5%: software documentation
- this is likely a featured article

Motivation

- Automatic detection of poorly written and poorly referenced articles will help editors to focus on the truly non-filled articles
- The differences in the same article across languages can be detected
- Possible output of important features should help Wiki-editors to concentrate on the most important gaps of the article

Work Pipeline



Data Processing

- Download wikipedia XML dumps
  - We are using page article multistream dumps. To get faster development loops so far we worked with a single dump, next we will run the full pipeline on the whole wikipedia data.
- Parse XML to CSV using streaming XML parser
  - We are using lxml and handwritten parser that goes through the file tag by tag and parses articles and meta information and article and last revision. The data we are fetching includes article text, title, revision author, revision comment and timestamp
- Fetch ORES assessments
  - Here we use mwapi and ORES web service to get article scores
- Inspect internal structure of text
  - Wikipedia articles have it's own syntax for declaring blocks inside article: source

FA	Professional, outstanding, and thorough; a definitive source for encyclopedic information.
A	Very useful to readers. A fairly complete treatment of the subject. A non-expert in the subject would typically find nothing wanting.
GA	Useful to nearly all readers, with no obvious problems; approaching (but not equalling) the quality of a professional encyclopedia.
B	Readers are not left wanting, although the content may not be complete enough to satisfy a serious student or researcher.
C	Useful to a casual reader, but would not provide a complete picture for even a moderately detailed study.
Start	Provides some meaningful content, but most readers will need more.
Stub	Provides very little meaningful content; may be little more than a dictionary definition. Readers probably see insufficiently developed features of the topic and may not see how the features of the topic are significant.

Feature Engineering

Were built features that reflect diversity of text, sources and links:

- Internal, external references count
- Internal, external references count / Number of paragraphs (Average number of references per block of text)
- Citations count and ratios (Journals, Books, Web, News, etc)
- Number of images, files, etc in the articles
- Number of non-approved references (“citation needed”)
- Headings count
- etc

```
df_features.printSchema()

root
 |-- sha1: string (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- title: string (nullable = true)
 |-- text: string (nullable = true)
 |-- Stub: double (nullable = true)
 |-- Start: double (nullable = true)
 |-- C: double (nullable = true)
 |-- B: double (nullable = true)
 |-- GA: double (nullable = true)
 |-- FA: double (nullable = true)
 |-- n_words: integer (nullable = false)
 |-- level2: integer (nullable = false)
 |-- level3: integer (nullable = false)
 |-- level4: integer (nullable = false)
 |-- level5: integer (nullable = false)
 |-- level6: integer (nullable = false)
 |-- book_citations: integer (nullable = true)
 |-- journal_citations: integer (nullable = true)
 |-- n_internal_links: integer (nullable = false)
 |-- n_external_links: integer (nullable = false)
 |-- n_paragraphs: integer (nullable = false)
 |-- n_unreferenced: integer (nullable = false)
 |-- n_categories: integer (nullable = false)
 |-- n_images: integer (nullable = false)
```

Modeling

Clustering algorithm: Bisecting K-means

The algorithm starts from a single cluster. Iteratively it finds divisible clusters on the bottom level and bisects each of them using k-means, until there are k leaf clusters in total or no leaf clusters are divisible.

+ hierarchical top-down approach
+ parallelism
+ high speed and efficiency (in terms of entropy, F measure and overall similarity)
- needs a hyperparameter k — fixed number of clusters — as input; but can be solved by maximizing the likelihood of the evaluation metrics

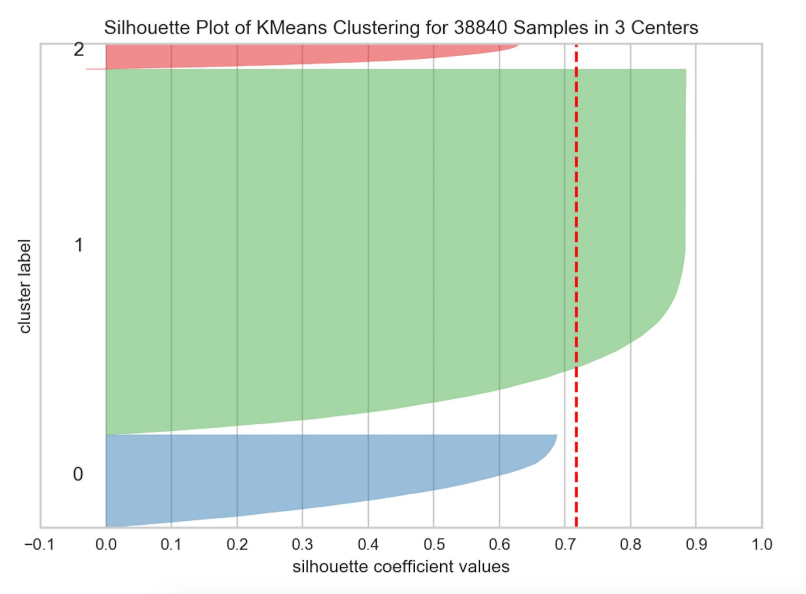
Evaluation

Silhouette coefficient measures how appropriately data have been clustered

It is calculated using

- mean intra-cluster distance **a**
- mean nearest-cluster distance **b**
- final score is **(b - a) / max(a, b)**
- score is between **-1** and **1**

for each sample



Current result (average for each cluster):

0.65; 0.68; 0.89

Further work

- Create supervised machine learning model with ORES as labels and our features as inputs. The resulted model should be transferred to the other languages that didn't support by ORES (like Ukrainian)
- Test the PySpark MLP / Random Forest / Gradient boosting and get the **features importance** (visualize the pluses and minuses of the articles references)
- Fix clustering with ORES features