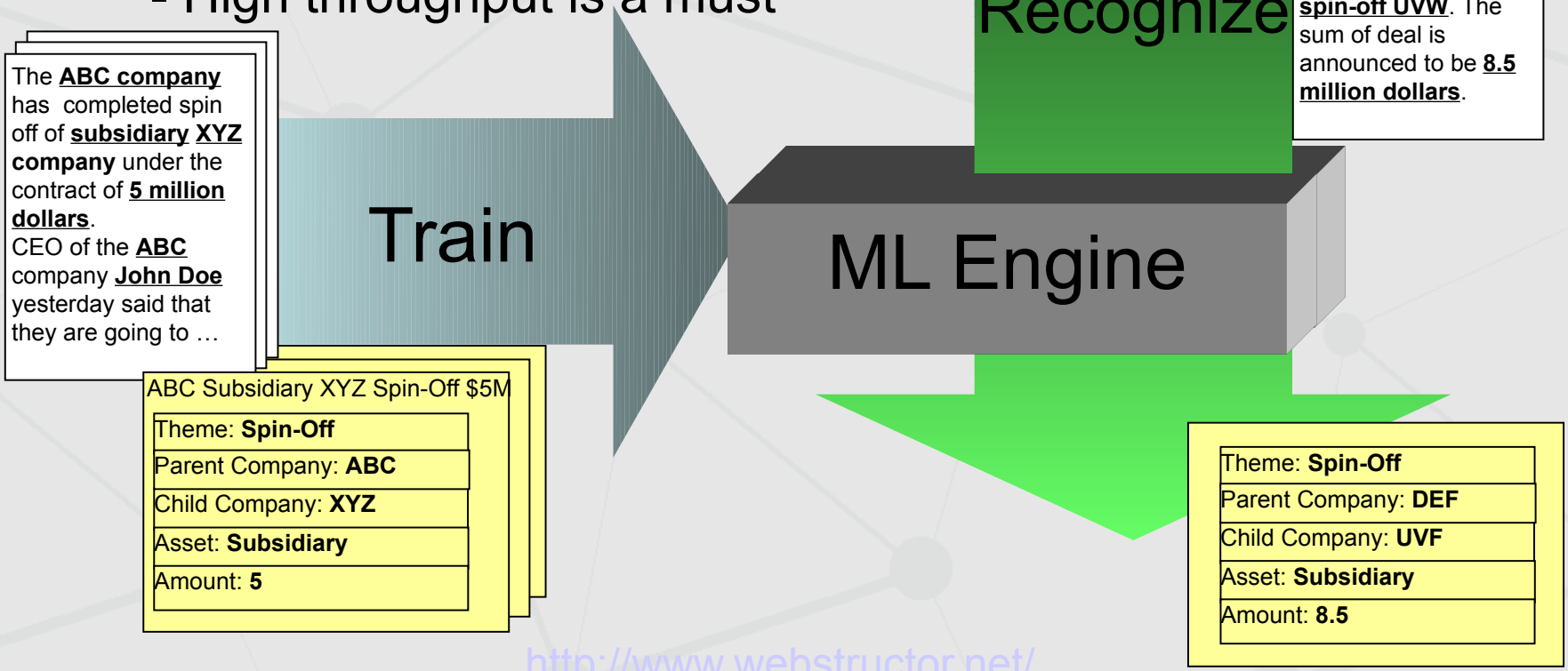
High-performance automatic categorization and attribution of inventory catalogs

Anton Kolonin
Webstructor project
2013, October

High-performance automatic categorization and attribution of inventory catalogs

The problem

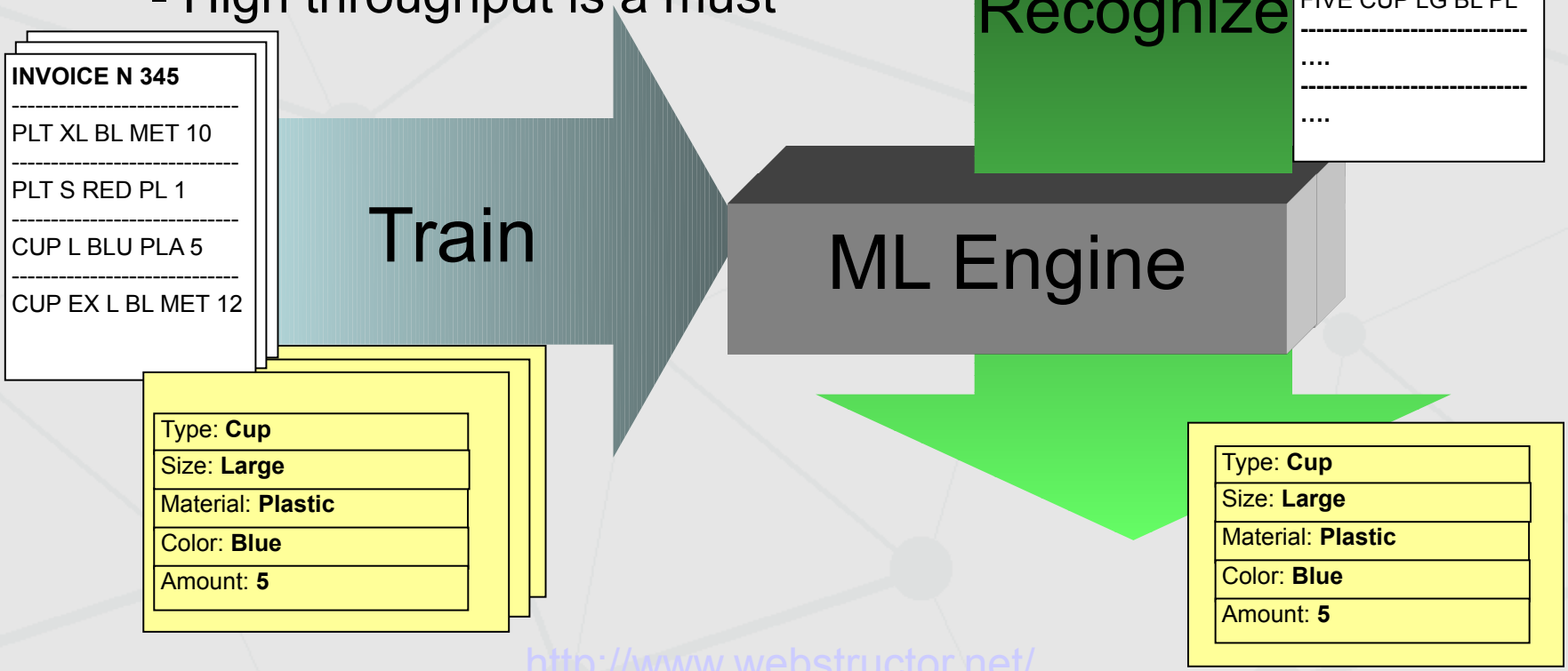
- Assign attribute values for item by its textual description
- Each item description contained in one text line
- Attribute values may be represented by multiple categories
- Each attribute correspond to specific domain of categories
- Heavily abbreviated texts
- Huge volume of data
- High throughput is a must



High-performance automatic categorization and attribution of inventory catalogs

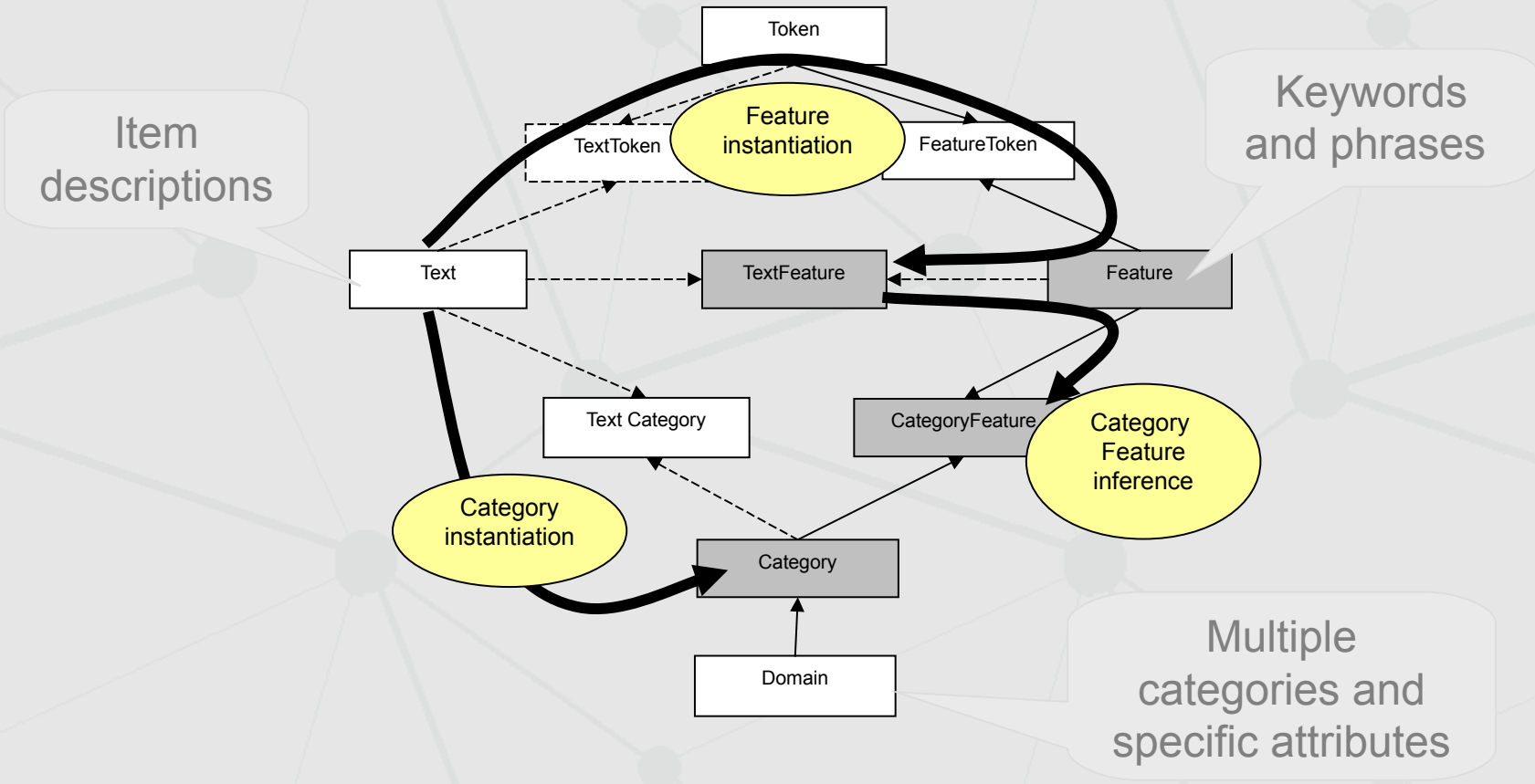
The problem made specific

- Assign attribute values for item by its textual description
- Each item description contained in one text line
- Attribute values may be represented by multiple categories
- Each attribute correspond to specific domain of categories
- Heavily abbreviated texts
- Huge volume of data
- High throughput is a must



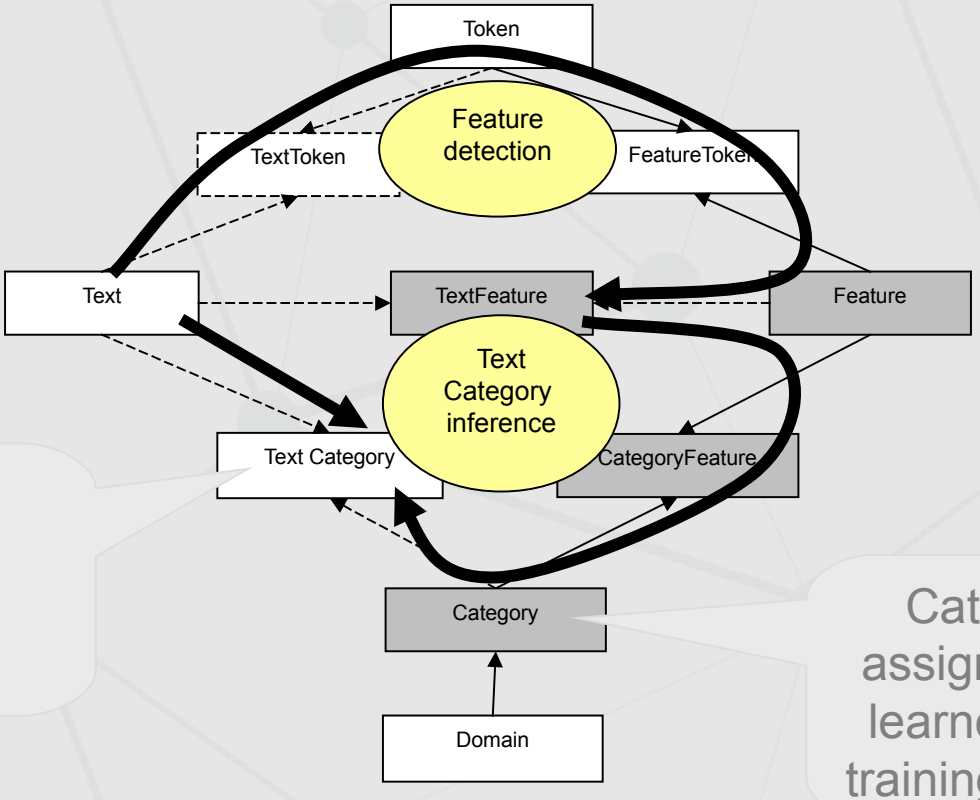
High-performance automatic categorization and attribution of inventory catalogs

Training Process



There are three sub-processes contributing to the learning process.
The **first process is Category instantiation** which takes the attributes defined for text in training corpus (either encoded in the text as tags or taken from respective database table fields) and creates categories for them, given the domain indicated by the attribute.
The **second process id Feature instantiation** which takes the text in training corpus and decomposes it into tokens and features accordingly to the parser, tokenizer and feature builder depending on the implementation.
The two processes above are independent, but they precede **the third process which is Category Feature inference**. It employs statistics to infer the relevance of features encountered in the texts to the categories associated with those texts.

Recognition process



Categories assigned to particular descriptions

Category assignments learned from training corpus

There are two sub-processes contributing to the rule applying process and the following process flow diagram depicts the dependency between the sub-processes and the data.
The **first process is Feature detection** which takes the text in novel data and decomposes it into tokens and features accordingly to the parser, tokenizer and feature builder depending on the implementation. This process is similar to Feature instantiation in the course of learning, but the key difference is that only the features instantiated earlier in the course of learning can be detected, no new features are instantiated.
The **second process is Text Category inference**. It employs statistics to infer the relevance of texts to the categories associated with those texts through the features detected in the texts and learned for those categories.

Options driving performance and accuracy

Technical options

- Programming language (C++, **Java**, ***Lisp***)
- Programming Platform (GNU Java, **Sun Java**, ***Franz Allegro Lisp***)
- Architecture (**in-memory** vs. database)
- Deployment (domain-specific server nodes, when possible)

Algorithm options

- Simple **SVM** model (with tuned weighting function)
- Learning model (**batch**, incremental)
- Features - "**keyword**" and "**keyword frame**"
- **Contextual scoping** (if not clustering processing nodes per domain)
    - Domains by category
    - Categories by category
- Restricting to **best categories per feature** (storage and run-time)
- **Priority on order** (rank "keyword frames" higher)
- **Boolean ranking** (based on count of features matching category)

High-performance automatic categorization and attribution of inventory catalogs

Impact of non-linear algorithm configurations on accuracy

MaxFrameDistance	**2**	**2**	**2**	**2**	**2**	**2**	**0**	**0**
MinKeywordLen	**2**	**2**	**2**	**2**	**1**	**1**	**1**	**1**
Best Categories/Feature	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
Heuristics	**None**	**Priority on Order**	**Boolean Ranking**	**Priority on Order and Boolean Ranking**	**None**	**Priority on Order and Boolean Ranking**	**None**	**Boolean Ranking**
Training set 1 (360000)	94.55%	95.36%	94.93%	96.07%	94.94%	96.52%	81.08%	76.28%
Training set 2 (645000)	96.97%	97.40%	97.23%	97.85%	97.16%	98.05%		
Test set 1 (18000)	85.35%	83.98%	84.71%	83.50%	85.23%	83.31%	78.84%	73.88%
Test set 2 (25500)	75.77%	74.37%	74.00%	73.29%	75.20%	71.07%	70.68%	54.24%
Test set 3 (73000)	79.74%	78.11%	78.27%	77.42%	79.79%	77.08%	74.72%	68.81%

MaxFrameDistance maximum distance between keywords used to build the ordered sets of keywords

MinKeywordLen minimum length of keyword in characters to involve keyword in the machine learning

Best Categories/Feature maximum number of possible attribute values associated with a keyword or ordered set of keywords to be considered

Heuristics:

None no heuristics involved, pure machine learning

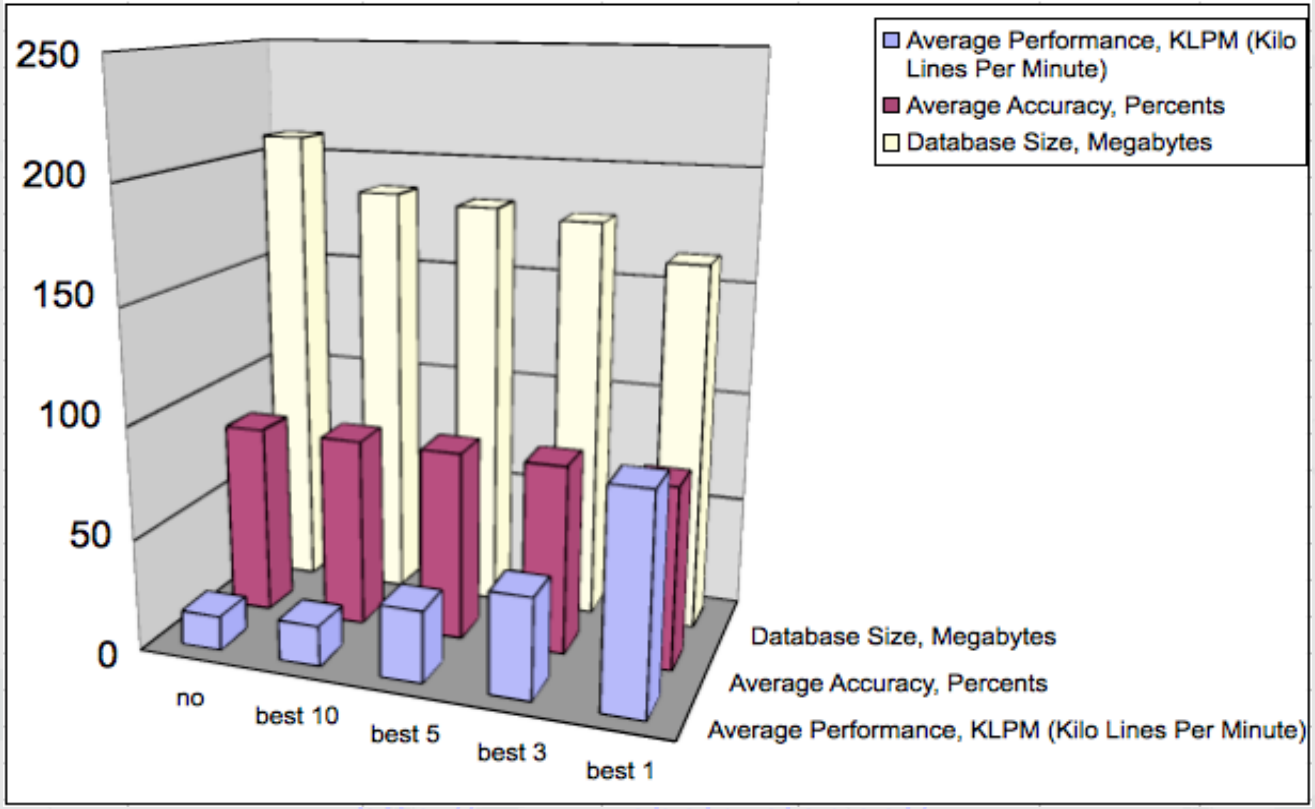
Priority on Order discern attribute value relying on the ordered sets of keywords first, and discern on the individual keywords only if the former fails

Boolean Ranking pay more attention to joint match of all keywords and ordered sets of keywords associated with an attribute value than to individual matches

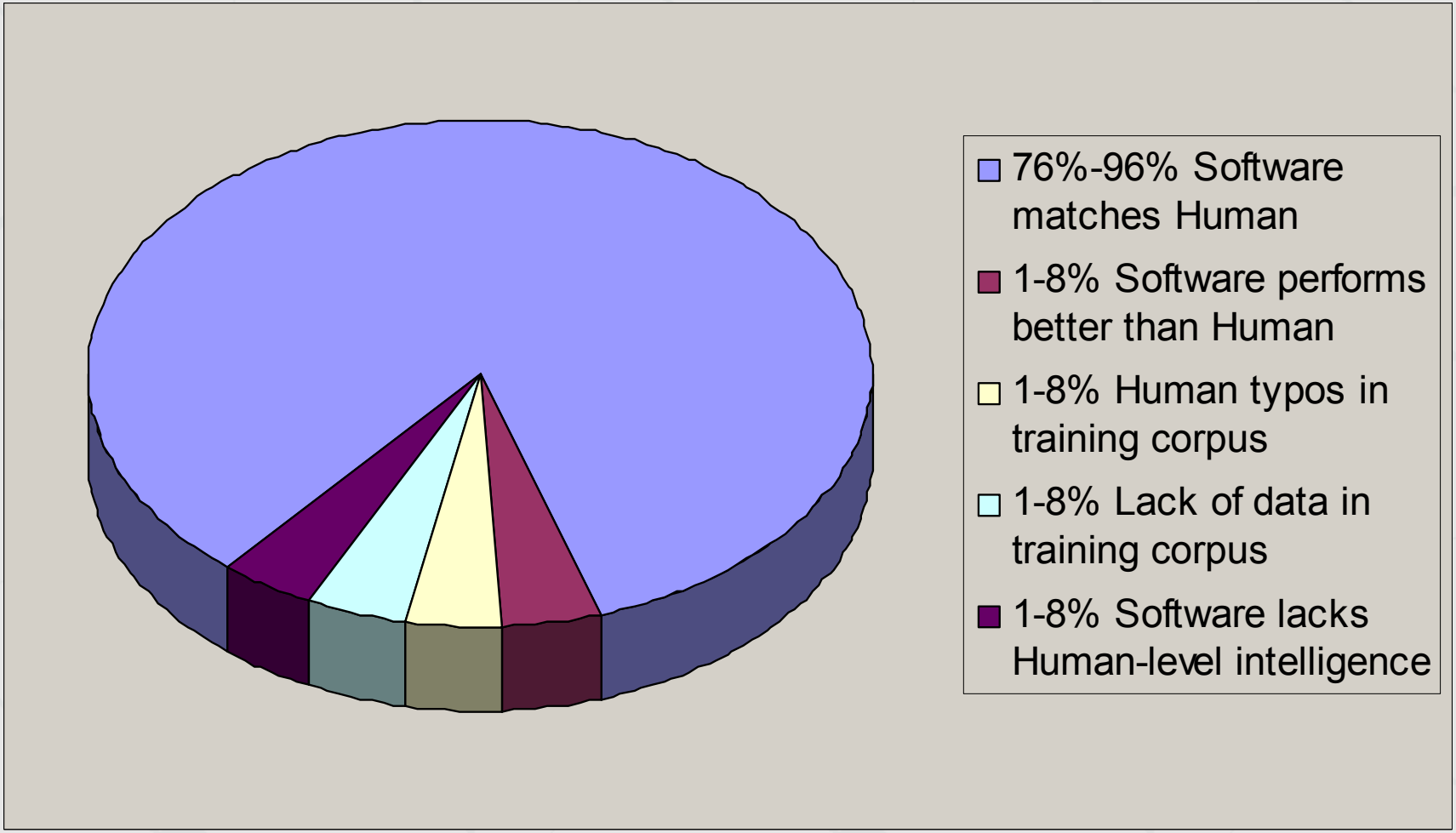
High-performance automatic categorization and attribution of inventory catalogs

Impact of best «categories per feature» on accuracy, capacity and performancy

compression cpf	Average Performance, KLPM (Kilo Lines Per Minute)	Average Accuracy, Percents	Database Size, Megabytes
no	15,28	85	208,932991
best 10	18,04	85	185,766692
best 5	32,04	85	181,901660
best 3	44,92	84	178,024282
best 1	95,32	81	162,849580



Comparing software results to human results



Thank you for attention!

Anton Kolonin
Webstructor project
http://webstructor.net/