# An Ensemble for Query Intent Detection
# CIKM CUP 2014, 3rd Place

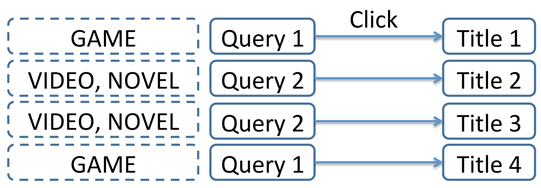
**Elmar Haussmann**
**University of Freiburg, Germany**
**Chair of Algorithm and Data Structures**

**CIKM CUP** sponsored by Baidu at
ACM International Conference on
Information and Knowledge Management,
Shanghai, China, 2014

## Task

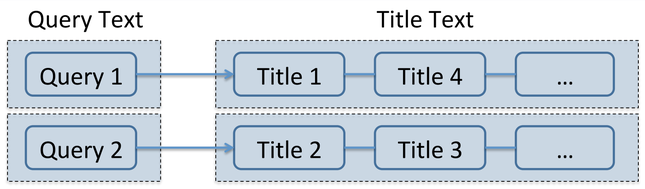
Determine "topics" of queries which can be one or more of 7 classes [1]:
VIDEO, GAME, NOVEL, OTHER, ... → **multi-label classification**

- Labels for 39k queries are provided
- Queries are structured in search sessions (from query log)
- Title of clicked link of query is provided (if any)




## Data Transformation

Create **documents** from queries and their associated clicked titles




- Representation is two-fielded document: query and concatenation of all clicked titles
- Introduce stop markers between clicks and at start/end for better features (n-grams)
- Only small number of queries without clicks: 170 w/o, 5k with fewer than
  → resonable representation
- Unlabeled queries (labeled "UNKNOWN") in provided data are ignored (incorporated in other features, see box on the right)
- Information about queries in same session is lost (incorporated in other features, see box on bottom-right)

## Text Classification Features

Treat problem of query intent classification as text classification

**Features from text**
- Unigram term counts
- N-grams counts of length 2 and 3 with minimum frequency 2 and 3, respectively
- To account for document length: use BM-25 [2] weighting (k=2, b=0.75)
- Note: query and clicked titles fields are seperate, i.e. an occurrence of the same term in a query and a title corresponds to different features

## Distributional Similarity Features

Utilize **distributional similarity** of words to create robust features

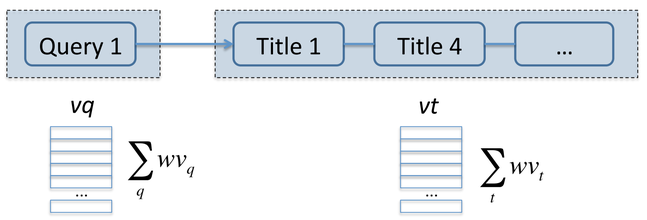
- Large amount of unlabeled data (queries labeled "UNKNOWN")
- A lot of characters/words occur only few times in labeled data (hard to learn from)

→

- Learn vector for each word representing its semantic
- Intuition: a word often appearing close to, e.g., "video words" in unlabeled data is probably also related to video

**Learning word vectors**
- Use all text of provided data (labeled and unlabeled, queries and clicked titles)
- Replace frequent n-grams (top 10k n-grams according to maximum likelihood ratio [4, chapter 5.4.3] ) by their concatenation
- Use Word2Vec [3] to learn real-valued vector for each word
- Train two models with different parameters (see below): one for title word vectors, one for query word vectors


**Combining word vectors**
- For a query with words *Q* and associated title text with words *T*, simple vector addition is performed to obtain a real-valued query vector *vq* and a title vector *vt*
- These vectors can directly be used as features of a query and/or its titles

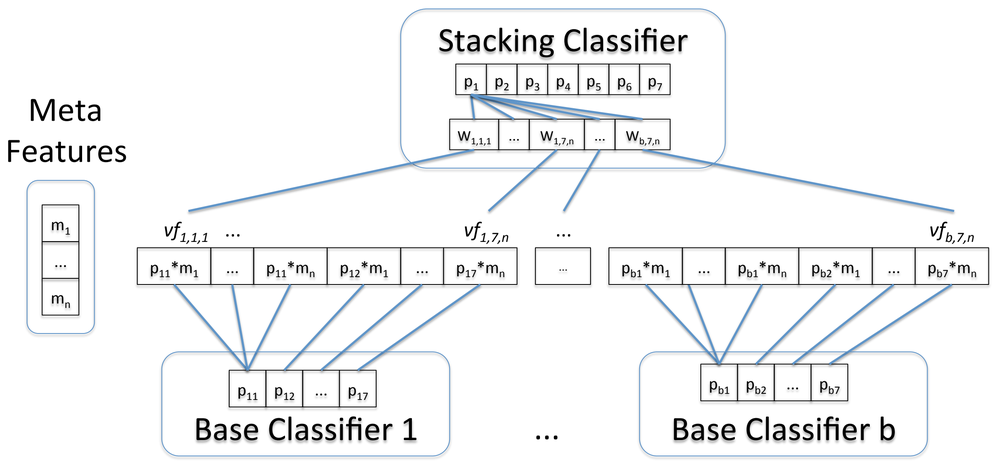



**Word2Vec parameters**
- Title vectors model: window=10, #dimensions=2000, min-count=5
- Query vectors model: window=5, #dimensions=1000, min-count=1

## Stacking Classifier

Train different *base* classifiers and combine their output using *stacking*

- Train different *base* classifiers on different features
- All learned base classifiers output probabilities and use Binary Relevance ("one-vs-all") for multi-label classification
- Combine predictions of base classifiers by training a final logistic regression classifier (l2-regularized) on their outputs: a form of **Feature Weighted Linear Stacking [5]**




**Above**
- Stacking Classifier for *b* base classifiers and *n* meta-features (and 7 classes) for a single example
- The input of the stacking classifier are products of each meta-feature extracted from the example and each base classifier output ("virtual features" $vf_{1,1,1}$ to $vf_{b,7,n,}$)
- For functions $m_j(x)$ that extract meta-feature *j* and $f_{b,i}(x)$ that correspond to the output for class *i* of classifier *b* on example *x*: $vf_{b,i,j} = f_{b,i}(x)m_j(x)$
- Additionally, original output probabilities are retained by adding an artificial meta-feature with value 1

**Meta Features**
- #of query text-classification features unigram
- #of query text-classification features n-gram
- #of query text-classification features in total
- Each of the above divided by their average in training set

**Base Classifiers**
- Logistic Regression L2-regularized
- Naïve Bayes
- K-Nearest Neighbors
- Random Forests

With different combinations of feature sources (only query, only title, both) and different features (text classification, distributional, session, query click graph) a total of:
**→ 21 base classifiers achieving 92.22 % F-1 on public leaderboard**

## Additional Features

**Session Features**
- For each query, count labels of other queries in same session
- But: many queries don't have other labeled queries in same session

→ Weak base classifier, still valuable in ensemble

**Query Click Graph Features**
- Create graph *G* with unique queries *Q* and titles *T* as nodes
- There is an edge *e* between *q* and *t* iff *t* was click for *q*
- The set of labels of a title *t* = set of labels of connected queries
- For each query count labels of directly connected titles

→ Weak base classifier, only miniscule improvement in ensemble

## Additional Information

**Learning and Model Selection**
- Split labeled training data into TRAIN and DEV (very conservative 50/50) for model/feature selection
- Train final classifiers train on TRAIN + DEV
- For training stacking classifier: train base classifiers with 20-fold cross-validation, i.e. in turn train on 19/20 apply on 1/20 held-out examples, use predicted probabilities on held-out examples to train stacking classifier
- **Best base classifier** (logistic regression + text classification features) achieved **91.52 %** on leaderboard; ensemble improved on that by (only) 0.7%

**Possible Improvements** (not followed due to time constraints)
- Automatic model selection for ensemble (i.e. which base classifiers to use)
- Deeper investigation of meta-features
- Better session and query click graph features

**Things That Did Not Work** (but should or might)
- Multi-label classifiers addressing label correlation
- Applying text segmentation approaches instead of using n-grams (e.g. Morfessor)

For questions: haussmann@informatik.uni-freiburg.de

## References

**[1] CIKM CUP 2014,** http://openresearch.baidu.com/activitycontent.jhtml?channelId=769 (2014)
**[2] Robertson, Stephen E., et al.**: Okapi at TREC-3. In: **NIST SPECIAL PUBLICATION SP** (1995)
**[3] Mikolov et al.: word2vec** https://code.google.com/p/word2vec (2013)
**[4] Manning, Christopher D.**: *Foundations of statistical natural language processing*. Ed. Hinrich Schütze. In: **MIT press** (1999)
**[5] David Lin et al.:** Feature-Weighted Linear Stacking. In: **arXiv:0911.0460** (2009)