Occupation Coding During the Interview

Appendix C: Description of the Algorithm

Numerous different algorithms exist to predict possible job categories. Our approach combines some of them to improve over any singular prediction. Schierholz (2014) provides additional background information for details not covered here.

We have different coding methods $m = 1, ..., M$ that calculate scores $\theta_{lj}^{(m)}$ for each response l and all possible job categories j. All the different scores are expected to correlate with the true probability, $P(c_j|l)$, that category c_j is correct for respondent l. We build for each respondent a data frame with $J = 11194$ rows for the different job categories. Each row j indicates that job category c_j could be correct. When for a person l the correct category is known from training data this is inserted into the data frame. This variable, "c_j correct", is the target variable we want to predict. All scores from the different models, $\theta_{lj}^{(m)}$, are also included into the data frame and serve as covariates. An exemplary representation of this data frame for one person is shown in table C1. Similar data frames are created for all other responses and attached to obtain a single large data frame. For computational speed we only keep those rows in the data frame where at least one score obtained via the verbatim answer indicates that this category could be correct. When the text does not indicate the possibility of correctness, the row is removed. For example, if the answer is "nurse", many health job categories are meaningful code suggestions but job categories from gardening and floristry are certainly incorrect. The resulting data set has 461,816 rows.

Gradient boosting is used with regression trees as base learner to predict the binary variable "c_j correct". This method was chosen because it allows flexible interactions between covariates and the correctness probabilities for c_j are calculated fast when a new response is entered.

Two challenges come with this approach:

- It is well known that predictions are biased when the same data is used twice in estimation (e.g., LeBlanc and Tibshirani, 1996; Breiman, 1996). In our case one would need the data a first time to estimate the functions that predict the scores $\theta_{lj}^{(m)}$

Table C 1: Exemplarious data frame for person l with correct job category $c_j = 01104101$

c_j	c_j *correct*	$Score_{lj}^{(1)}$	$Score_{lj}^{(2)}$	$\cdots$
$c_1 = 01104100$	FALSE	$\theta_{l,1}^{(1)}$	$\theta_{l,1}^{(2)}$	$\cdots$
$c_2 = 01104101$	TRUE	$\theta_{l,2}^{(1)}$	$\theta_{l,2}^{(2)}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$c_{11194} = 99998115$	FALSE	$\theta_{l,11194}^{(1)}$	$\theta_{l,11194}^{(2)}$	$\cdots$

and a second time to estimate the global boosting model. To avoid double usage, the prediction models for initial scores $\theta_{lj}^{(m)}$ do not use observation l for their estimations ("stacking").

- Computers need to have very large RAM to load a boosted model when the training data consists of many observations. In our case, it was not possible to make predictions with the complete training data. Instead, we split the data at random by rows in ten disjoint sets and estimate five separate boosting models, leaving the other five sets aside due to performance restrictions. To predict a new code for a given response l we can then (1) predict the scores $\theta_{lj}^{(m)}$ from all training data, (2) predict probability vectors for "c_j correct" with each of the five different boosting models, and (3) average over the five predictions.

We next explain how the different scores $\theta_{lj}^{(m)}$ are calculated. Each entry in the following list corresponds to one predictor in the boosting model.

- Number of (partial) dictionary matches from the alphabetic dictionary that suggest category c_j. A match is found when either the input text or a substring of it is identical to a preprocessed job title from the database. The dictionary is the "Berufs- and Tätigkeitsverzeichnis" that is published online.[1] (+ phrase)
- Number of dictionary matches from the search word dictionary that suggest category c_j. A match is found only when the input text is identical to the technical name. The dictionary is the search word file B_SW.txt that is published online.[2] (+ phrase)
- Number of (partial) dictionary matches from the search word dictionary that suggest category c_j. A match is found when either the input text or a substring of it is identical to a preprocessed job title from the database. The dictionary is the same as before. (+ phrase)
- Number of identical answers in ALWA training data that were coded into category c_j (only exact matches). (+ phrase)
- Number of (partially) identical answers in ALWA training data that were coded into category c_j. A match is found when either the input text or a substring of it is identical to the training data.
- Posteriori expectation for the probability that category c_j is correct. Schierholz (2014) describes the underlying Bayesian model. (+ phrase)
- Posteriori probability that the probability for category c_j is larger than 0.05. This number comes from the same Bayesian model as before. (+ phrase)
- The likelihood to observe the input text for every possible category,

$$P(input\ text|c_j) \propto \prod_{v=1}^{V}(0.95 * P(T_v|c_j) + 0.05 * P(T_v))$$

[1] http://statistik.arbeitsagentur.de/Navigation/Statistik/Grundlagen/Klassifikation-der-Berufe/KldB2010/Systematik-Verzeichnisse/Systematik-Verzeichnisse-Nav.html

[2] http://download-portal.arbeitsagentur.de/files/

where the product is over all terms T_v that appear in the input text. $P(T_v|c_j)$ and $P(T_v)$ are both relative frequencies as calculated from the training data. $P(input\ text|c_j)$ is standardized to sum to 1. A motivation for this model, which is based on the Naïve Bayes assumption, is given by Schierholz (2014). (+ phrase)

- For closed questions we calculate the likelihood for the respondent's answer given all possible categories, $P(answer|c_j) = \frac{\#\{answer, c_j\}}{\#\{c_j\}}$. If the likelihood is smaller than 0.03 it is set to 0.03. Likelihoods are calculated for each of the following questions and used as covariates in the boosting model:
  - occupational status (question number 6.2)
  - differentiated occupational status (question numbers 6.2 & 6.7 - 6.10)
  - For self-employed persons the number of staffers (6.13)
  - Does a person have management responsibilities? (6.14)
  - If so, for how many employees? (6.14+6.15)
  - Which education is usually required for your job? (6.16)
  - Industry (6.17)
  - Number of employees in the company (6.18)

  The count variables 6.13, 6.15 and 6.18 are aggregated to form discrete variables.
- Multiplying all the likelihoods $P(answer|c_j)$, $P(input\ text|c_j)$, and a prior $P(c_j)$ provides, after standardization, an estimate for $P(c_j|respondent\ l)$.
- An indicator is set to 1 if the phrase is not identical with the complete preprocessed verbatim answer, 0 otherwise.
- Several scores are calculated from the complete preprocessed verbatim answer. An alternative is to run the same algorithms but use only the phrase as input text instead of the full answer. Seven additional covariates for our boosting model are calculated doing exactly this. The respective items in this list are marked with "(+ phrase)" above.
- A last covariate counts for how many categories there exists an indicator that this category could be correct for person l. The possibility of correctness is indicated if at least one of the first five entries from this list is larger than zero, either when the complete verbatim answer is used as input or the phrase.

Interviews were carried out with the `NIPO fieldwork system` (NIPO Software, 2014) that writes relevant data to a `MySQL` database (Oracle Corporation, 2014). At the same time, NIPO launches the statistical programming language `R` (R Core Team, 2012) that processes the data and writes the suggested categories for the respondent back to the database. NIPO, in turn, reads the new database entries and shows the suggested categories to the interviewer. Because the algorithm requires more RAM than what is available on typical interviewer computers, calculations were carried out on an external server. For data handling in R, we employed the packages `Rserve` (Urbanek, 2013), `foreign` (R Core Team, 2014), `data.table` (Dowle et al., 2014), `tm` (Feinerer et al., 2014), `stringr` (Wickham 2012), and `RODBC` (Ripley and Lapsley, 2013).