Learning a Semantic Representation of Atomic Entities for Salary Prediction

Keith Chewning
Burning Glass Technologies
Boston, MA
kchewning@burning-glass.com

Zhiyuan Liu
Burning Glass Technologies
Boston, MA
zliu@burning-glass.com

Manish Gaurav
Burning Glass Technologies
Boston, MA
mgaurav@burning-glass.com

ABSTRACT

Salary is an essential component of any job. Job candidates indicate that they would like to see salary more than any other feature in a job posting.[1] Intuitively, people have linear expectations about salary. For instance, a person with more experience should earn a higher salary than a person with less experience. Given the importance of salary and peoples' intuition about salary, a salary modeling task must understand the semantic nature of the input features in order to produce predictions sufficient to one's expectations. We will detail the model selection process and relative model performance of deep neural networks built for this task and how feature weighting is incorporated into these models. We will then look at evaluation of the model both from a quantitative and qualitative view. This qualitative evaluation will show the network's understanding of semantic structure inherent in the input data. Finally, we introduce a novel method to jointly gain insight into feature importance of the artificial neural network and to perform variance-reducing model averaging, a form of auto-ensembling.

KEYWORDS

Deep Learning, Artificial Neural Networks, Semantic Learning, Regression

1 INTRODUCTION

Burning Glass Technologies (BGT), founded in 1999, is an analytics software company that continually collects job posting data from more than 45,000 online sources. Job postings are obtained from online job boards and company websites through BGT's spidering technology. The BGT database consists of more than 700 million job listings. Approximately 80% of all collected postings are determined to be duplicates and discarded. This ensures that the BGT database contains unique postings. Each job posting is parsed to extract more than 50 unique data elements. Additionally, there is a detailed BGT taxonomy which includes occupational and skill hierarchies. Robust and nuanced salary data, like the BGT dataset, is an important part of any labor market analysis task.

Utilizing the BGT job posting dataset, there are a number of applications which we planned to address in the development of this model.

(1) Provide predicted salaries for all job postings in the BGT database.
(2) Aggregate salary information based on parsed and tagged features.
(3) Predict salary for a "custom job" based on user input features.
(4) Provide insight into the relative importance of the input features to the model.

[1]A 2016 CareerBuilder survey [2] indicates that 74% of candidates want to see salary more than any other feature.

Only about 15% of BGT job postings have parsed salary information. Given that salary is a very important feature [2] of a job posting, we want the ability to provide predicted salary information for every job posting in the database. Item (2) above is a product-level application of Item (1). Given salary predictions for all job postings, aggregate statistics are simply SQL queries over pre-computed predictions. While straightforward, we cannot assume that this methodology will produce reasonable results. So, this is something that needs to be evaluated.

Item (3) above provides users the opportunity to generate salary predictions in real-time, not by querying pre-computed predictions over existing job postings. In particular, users may type in titles and select taxonomic features to generate salaries for jobs with unique features. In order to accomplish this, the model needs to properly generalize. Due to data sparsity, it is likely that user generated job features could be distant from what the model observed during training. The generalization of the model must be tested to observe the model under these conditions.

Item (4) is meant to provide insight into the relative importance of the input features. Neural network models are notoriously considered "black-box" models. A novel approach is considered to gain insight into how to interpret an input features' influence on predicted salary.

While addressing the above items the following are main contributions of this paper.

(1) To build a highly generalizable deep neural network model which marries both semantic learning and predictive accuracy.
(2) To explore and promote a diverse set of model testing criteria. Here we introduce *qualitative* analysis which is critical to having a comprehensive performance evaluation of the model.
(3) To develop a methodology, usable at prediction time, which jointly provides insight into relative feature importance at the record level and to improve the predictive capacity of the model.

The development phase of the salary model, which leverages BGT data, and its evaluation is complete. Engineering effort for production deployment is currently in-progress and release of the system is scheduled at the end of the first quarter, 2018.

2 RELATED WORK

In 2013, Kaggle, an online platform for predictive modeling competitions, hosted a competition for Adzuna [5] to predict salary directly from job posting text. The winner of this competition, Vlad Mnih, finished with a mean absolute error (MAE) of 3464. Judging from the text of the job postings, the most probable conclusion is that the currency of the salary in this competition is British Pounds. Using an exchange rate of 1.56 [7], which is a reasonable exchange rate at the time of the competition, this MAE would equate to about 5403 USD. For comparison, we can use this a rough baseline for comparing the MAE or our model. This competition expected, as input into the model, the full job text. The model proposed here leverages parsed features from a job text as input into the model and does not consider the full job text. The reason for this decision is to allow for all of the applications stated above. User selected skills and taxonomy features would not be possible by conforming ourselves to raw job text.

Both LinkedIn [9] and Stack Overflow [6] now offer salary information. LinkedIn is concerned about user anonymity and takes great pains to enforce this requirement. Their salary model predicts user earnings where the BGT model predicts posted salaries. Presumably these are very similar, but they are not exactly the same thing. LinkedIn salary information is based on *cohorts*, which aggregate data by occupation and location (and potentially other features). The BGT model on the other hand seeks a granular per-posting view which discerns the individuality of a particular posting.

The Stack Overflow Salary Calculator is based on a survey of 64000 software developers. Similar to LinkedIn, Stack Overflow shows salaries based on aggregates (e.g. rôle and location), but they do provide an extra layer of granularity by incorporating skills. The most prominent difference between the Stack Overflow Salary Calculator and the BGT salary model is that Stack Overflow operates on a very narrow occupational window of just software developers. The BGT salary model is designed to model full spectrum occupations in the US labor market.

3 DATA

In the Burning Glass job postings database, each posting is parsed to extract and code more than 50 unique data elements. This parsing process extracts information from the job posting, like employer, location, and credentials. BGT places particular care in coding occupations and industries. Occupational coding consists of ascertaining entities from the BGT taxonomy, which is more granular and flexible then government occupation codes in order to adapt quickly to emerging markets. Occupational coding for postings also includes government occupational taxonomies like O*NET and SOC[2] codes.

The salary model described in this paper leverages these extracted entities and coded fields as input into the model. Specifically, we only use a small subset of these features:

- Title: title of the posting,
- Skills: relevant skills to the posting,
- MSA: metropolitan statistical area,[3]
- NAICS: North American Industry Classification System,[4]
- Employer: employer name,
- Education: educational requirements,

[2]The Standard Occupational Coding is a federal occupational coding system. More information is located as https://www.bls.gov/soc/. The more granular O*NET-SOC taxonomy is located as https://www.onetcenter.org/taxonomy.html.
[3]https://en.wikipedia.org/wiki/Metropolitan_statistical_area
[4]https://en.wikipedia.org/wiki/North_American_Industry_Classification_System

Table 1: Occupational Taxonomy

Hierarchy	Examples
CareerArea	Information Technology, Finance, Sales
OccGroup	Visual Design, Software Development
BGTOcc	K–12 Teaching, Database Architects
SubOcc	Air Traffic Controller, Python Developer

Table 2: Skills Taxonomy

Hierarchy	Examples
SkillClusterFamily	Human Resources, Legal, Health Care
SkillCluster	Advertising, Big Data, Solar Energy
Skills	Biochemistry, Forklift Operation

- Experience: years of experience, and
- OccGroup: occupational group.

All input features are optional. All fields may be user-defined, with the exception of OccGroup, which is inferred from the title prior to input into the model (if available). This occupational group label is used to provided input feature weighting, which will be described below. Title is the only *free text* field and all other features are taken as *atomic entities*.

The BGT taxonomic database leveraged here includes occupation and skills data. The occupational hierarchy is of the form

$$CareerArea \rightarrow OccGroup \rightarrow BGTOcc \rightarrow SubOcc.$$

CareerAreas are the highest level of occupational grouping in the BGT taxonomy. SubOccs (sub-occupations) are specialized occupations and provide a more nuanced view of jobs relevant for businesses and mid-career professionals. The next level of occupational granularity is job posting title, which is exempt from the BGT taxonomy. See Table 1 for occupational taxonomy examples. For input features, we only consider OccGroup from the occupational taxonomy, which is a feature of the model. Later we will reference SubOcc as a means of aggregating results.

The skill hierarchy is of the form

$$SkillClusterFamily \rightarrow SkillCluster \rightarrow Skill.$$

See Table 2 for skills taxonomy examples. The salary model only considers skills from this taxonomy.

The data used for this model was split into train and test sets. The training set consisted of 5M job postings from years 2016 and 2017. The test set is a 2.5M set from 2017. About 15% of the BGT dataset has parsed salary information. This data was further subset to retain only records whose salary fell with a given salary range per SubOcc. These ranges were defined based on historical data aggregated by SubOcc; outlier gold salaries falling outside the 5th and 95th percentile aggregates were excluded from training. This final set of data is what comprised the 5M/2.5M split.

Data observed during training typically had most fields populated. Given a user-defined input as suggested in Item (3) of the Introduction, a typical input might look like the following:

- OccGroup: Human Resources Specialists

- Title: Human Resources Administrative Assistant
- MSA: Bennington, VT
- Experience: 4 years
- Skills: Administrative Support, HRIS, Benefits Administration, Travel Arrangements, Office Supply Ordering, I-9 Audits, Payroll Processing, Data Entry, Onboarding, Mail Distribution, Calendar Management, Telephone Skills

The predicted salary for a rôle with these features is 39,360 USD. All salaries, both gold and predicted, are annualized and in USD. Hourly wage jobs are also annualized so that we are predicting values in the same units, USD/year.

4 FEATURE EXTRACTION

The features for this model create an extremely sparse input. Given that (free text) titles are tokenized, this alone could produce an infinite number of possibilities. This, in addition to all possible taxonomic inputs creates a very large feature space. A large feature space will produce a bottleneck for efficient learning. In order to compensate for this we leverage Scikit-learn's HashingVectorizer [13], which utilizes the "hashing trick."[5] This reduces the dimensionality of a feature space to a fixed size, which is accomplished by hashing each input feature. The value of the hashed feature is used as its column index in the new *hashing space*.

We refer to the raw text input into the model as features. These text features are transformed into numeric data prior to sending data to the model. This numeric input we refer to as its *parameters*, generic values in hashing space. This feature to parameter transformation goes through a number steps. The input features are bisected into two categories—weighted and unweighted. The weighted features are title and skills. All other features are unweighted. Unweighted features are assigned a value based on their frequency count. For the weighted features, the title is tokenized and skills are taken as atomic entities. Their weights are determined by a modified TF-IDF weighting scheme, where we utilize ICF or inverse category frequency as detailed in [16].

Independent of the model, but based on the training data, we formulate a sparse TF-ICF category-term matrix, M. The dimensionality of this matrix is the number of OccGroups by the dimensionality of the hashing space. We concatenate all skills and titles by OccGroup; this represents one category (row) in M. The scalar M_{ij} is the TF-ICF value of OccGroup i and hashed feature j. To be concrete, we may be given a job posting whose OccGroup is *Business Intelligence* and a particular skill in this posting is *Python*. If the index of *Business Intelligence* is row 98 and the hashed index of *Python* is column 41, then $M_{98,41}$ is the weight assigned to *Python* for this posting. The effect of this is to down-weight highly prevalent inter-OccGroup skill and title words. In the *Business Intelligence* OccGroup, we would not want to give a high weight to *Microsoft Excel*, this cross-cutting skill would be just as prevalent in an administrative function. If we had *Tableau Software*, however, this would be much more relevant to this OccGroup and should likewise receive a higher weighting. So, a particular skill or title word will have different weighting per OccGroup. The matrix weights are created in the following manner. Here, a term t is a hashed feature, a category c is an OccGroup, and the corpus is C, where c is in C.

[5] This technique was popularized by Vowpal Wabbit [10].

The weights of M are determined by the TF-ICF formula

$$\text{tficf}(C, c, t) = \text{tf}(c, t) \times \text{icf}(C, t)$$

where $\text{tf}(c, t)$ is the count of term t in c and

$$\text{icf}(C, t) = \log \frac{1 + |C|}{1 + \text{cf}(C, t)} + 1.$$

The value $\text{cf}(C, t)$ is the count of term t in across all categories in corpus C. So, $M_{ij} = \text{tficf}(C, i, j)$. We further scale the nonzero values of M with an empirical weighting formula to constrain its range to $[1, 3]$ with the formula

$$M_{ij} \leftarrow \frac{\log(\min(\max(1, M_{ij}), 2000))}{4} + 1.$$

As stated previously, the intuition here is that prevalent inter-OccGroup features will have diminished weights due to its ICF penalty.

We now have two input parameter matrices, weighted and unweighted of equal dimensionality. To merge these two parameter matrices, we take the element-wise maximum to derive the final input matrix. This numeric *parameter matrix* is the input into the model.

5 DEEP NEURAL NETWORK MODEL

The model used here is a deep neural network model built with the Keras [3] neural network API. As a baseline, we use a feedforward neural network model as proposed in [12]. The dimension of the hashing space is 2^{15}–the dimensionality of the input layer. There are three hidden layers of dimensions 1000, 300, and 300. For the selected model there is an $L1$ weight regularization, with a factor of 0.01, on these layers. The final layer is a single output node for the regression estimation. The activation function of the hidden layers is a Leaky ReLU with an α of 0.2. Prior to each activation we use batch normalization and no bias for these layers as proposed in [8]. The final output layer is a linear layer. The Adam optimizer is used and we optimize with the mean square error (MSE) loss function. We also evaluate the mean absolute error (MAE) of the model to have an evaluation metric in USD. See Algorithm 1 for the model pipeline. The Dropped Feature Analysis discussed below leverages `get_salary` by serial execution of this function.

5.1 Baseline Model and Model Selection

As an initial model we developed a three hidden-layer feedforward network with no regularization and no feature weighting. This model had a test set MAE of 4599. The base model as shown in Table 3 incorporates feature weighting as described above. This model had a test set MAE of 4216. All subsequent models utilize this same feature weighting. For comparison, the MAE of the model for the Adzuna Kaggle competition was 5403 (currency adjusted to USD). While not an exact comparison because of differences in input to the models, this does provide some context for evaluation of MAE. Given that neural network models are high variance models, we were concerned with overfitting of this model. We built 15 additional models with different configurations as detailed in Table 3 to address this concern. Models 2 and 3 implemented a dense residual network as proposed in [4], which provided some benefit. Extremely deep residual models were not tested. Model 3 uses a dual node output layer. The training data consists of a salary

Algorithm 1: Model Pipeline

```
Data: raw input feature set $F$, training data $T$, list of unique
      OccGroups $G$, $feature\_dim = 2^{15}$
Result: predicted salary
def weight_matrix():
    # group and concatenate all title tokens and
      skills of T by OccGroup
    $T_{OccGroup}$ := group_concat($T$);
    # hash to size $feature\_dim$
    $T_{hash}$ := hash($T_{OccGroup}$);
    # category-term matrix
    $T_{ctm}$ := ctm($T_{hash}$);
    $T_{tficf}$ := tficf($T_{ctm}$);
    # rescale weight matrix
    $M$ := scale($T_{tficf}$);
    return M;
def get_salary(F, M):
    $o$ := occgroup($F$);
    $g$ := index_where($G = o$);
    $u$ := unweighted_features($F$);
    $w$ := weighted_features($F$);
    $u_{hash}$ := hash($u$);
    $w_{hash}$ := hash($w$);
    $U$ := tf($u_{hash}$);
    $W$ := zeros($feature\_dim$);
    # update W with nonzero indices of $w_{hash}$
    $W$ := $M$[$g$, nonzero($w_{hash}$)];
    $P$ := elementwise_max($U$, $W$);
    # neural network net, predicted salary $\hat{y}$
    $\hat{y}$ := net($P$);
    return $\hat{y}$;
# One-time initialization
$M$ := weight_matrix ();
# Prediction
prediction := get_salary ($F$, $M$);
```

range—a minimum and maximum salary. This model is trained to learn both of these quantities. The interesting thing to note here is that the MAE for the minimum salary is greater than that of the maximum salary, which implies that there is greater uncertainty in the lower bound of the salaries. For all other models, the gold salary value is the mean of the minimum and maximum salary for a particular posting. Models 4 and 5 were a comparison between $L1$ and $L2$ regularization. Here, $L1$ regularization showed improvement. Compared to the base model, we can also see the effect of adding regularization to the model.

As discussed in Item (3) of the Introduction, models 6–10 were meant to address a certain applications of the model. It is possible that the parsed features observed during training are of a different distribution then what we may see at prediction time. In particular, a user may request a salary with just a title and one or two skills. This extreme sparsity of input is different from the training data where for most postings, many of the fields are populated. So, these models were meant to see how the model would be effected by removal of features. Model 6 implements *input dropout,* where we randomly drop 30% of the input features. This is not to be confused with standard dropout, which was utilized in model 10. Models 8 and 9 completely remove the specified features, where *Edu* is education and *Exp* is experience. As expected, the MAE of these models is higher but this penalty is not prohibitive, which implies that even with a sparser input there is still a signal in the other fields by which we may create a reasonable prediction. The worst-performing of these models is model 10 which utilized dropout. The rationale for this behavior is described in [11], which indicates that using dropout with sparse input is equivalent to reducing the amount of training data.

Model 12 was trained to confirm we were not overfitting the model with too many model parameters. This is a two hidden-layer network, which was the worst performing model of any that were trained, indicating that the model is not over-parameterized. The rest of the models test different regularization in conjunction with batch normalization. The best performing model 15, which is the base model with the addition of batch normalization and $L1$ regularization. The MAE of this model is 3844, an 8.8% MAE improvement when compared to the base model. In all models, early stopping was employed to avoid overfitting. At the epoch where test error started to bottom out, this checkpoint was used for the evaluation metrics in Table 3.

6 RESULTS AND EVALUATION

We now present a comparison between the base and final model and evaluation of the final model. For simplicity, we will refer to the final model as *model15.* MAE is one view of model performance. The base model has an MAE of 4216 and model15, 3844. This metric provides a global measurement of the model; on average, a predicted salary may deviate from the gold salary by about 3844 USD. While this shows some comparison between models, the reason for building new models was to reduce variance and improve generalization. To make this comparison, we look at a more granular view to see if there is variation in model performance at this level.

6.1 Normalized MAE

Observing MAE at the SubOcc level (the most granular BGT occupation classification) may skew the interpretation of this metric. An absolute deviation 2000 USD for a *Hostess*, where the gold salary $25000, is of a different relative scale than a 2000 USD deviation for a *Mechanical Engineer*, where the gold salary is $60000. In order to have a scale-invariant metric across SubOccs, we normalize these values.

For a given dataset $\mathcal{D} = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$, y_i is the gold salary for a particular posting and $\hat{y}_i = f(X_i)$ is its predicted salary. We may partition $\mathcal{D}$ by SubOcc into a set of mutually exclusive sets

$$\mathcal{D}_j = \{(X_j, y_j) \mid j \in SubOcc\},$$

Table 3: Model Comparison

Model #	Notes	L1	L2	Input Dropout	Batch Normalization	Dropout	Skills	Edu	Exp	Epochs	MAE
0	∗						✓	✓	✓	15	4599
1	★						✓	✓	✓	15	4216
2	†		✓				✓	✓	✓	15	4109
3	‡		✓				✓	✓	✓	15	4630/4098⋄
4			✓				✓	✓	✓	15	4776
5		✓					✓	✓	✓	10	4607
6		✓		✓			✓	✓	✓	10	4711
7		✓		✓	✓		✓	✓	✓	10	4651
8								✓	✓	10	4798
9										10	4942
10					✓	✓	✓	✓	✓	10	5201
11					✓		✓	✓	✓	10	4156
12	□	✓					✓	✓	✓	11	5348
13			✓		✓		✓	✓	✓	10	4315
14					✓		✓	✓	✓	9	3865
15		✓			✓		✓	✓	✓	12	**3844**
15a	○	✓			✓		✓	✓	✓	12	**3553**

∗	no TF-ICF feature weighting
★	base model
†	residual model
‡	residual model; dual output
□	two hidden layer model
⋄	minimum/maximum salary
○	Dropped Feature Averaging of model15

Table 4: Model Comparison of MAE_{share} Distribution Across SubOccs Here, 90% of the base model SubOccs' MAE_{share} is less than 19% and for model15, 15%. Smaller is better.

	MAE_{share} Percentile				
Model	10th	25th	50th	75th	90th
base	**8%**	10%	15%	15%	19%
model15	9%	**10%**	**11%**	**13%**	**15%**

Table 5: Model Comparison of the Distribution of Absolute Deviation Across SubOccs Here, 10% of the base model SubOccs' absolute deviations is less than 562 and for model15, 362. Smaller is better.

	Percentile				
Model	10th	25th	50th	75th	90th
base	562	1265	3546	7977	14893
model15	**362**	**922**	**2760**	**6598**	**12974**

the records in $\mathcal{D}$ which are tagged as the given SubOcc. The mean absolute error within a SubOcc is defined by

$$MAE = \frac{\sum_{j \in SubOcc} |\hat{y}_j - y_j|}{m},$$

where the number of records in a given SubOcc is m. We also let the mean gold salary over $\mathcal{D}_j$ be $\bar{y}$. Then we have

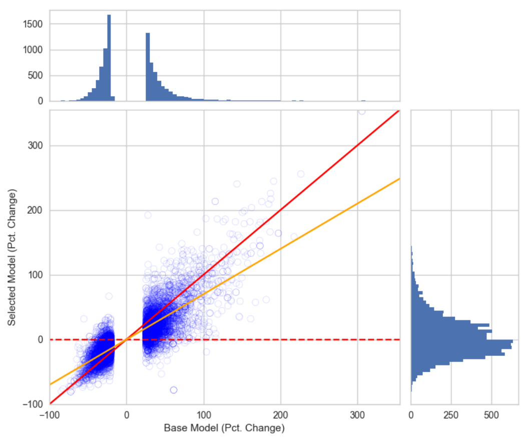
$$MEA_{share} = \frac{MAE}{\bar{y}}.$$

The rationale for this metric is that MAE may scale differently for different salary ranges. By defining the unitless metric MAE_{share}, we have a comparable way to measure deviation from the gold salary across SubOccs and hence varying salary ranges.

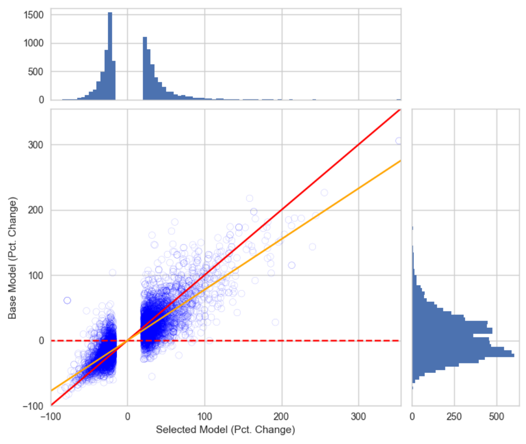
Table 4 shows a comparison of the base model to model15 looking at MEA_{share} over all SubOccs. We see, for example, that the 90th percentile MAE share reduced from 19% to 15% between models. This indicates that MAE is improved both at the global level and at the SubOcc level. So it does not appear that some SubOcc's MAE were adversely effected when comparing MAE in this manner.

6.2 Deviation Analysis

If model15 reduces variance and has better generalization relative to the base model, then we should see a compression of deviations from the gold salary between models. Table 5 looks at absolute deviations of the predicted salary from the gold salary aggregated across SubOccs. This is for a hand-selected set of 20 SubOccs determined by Burning Glass domain experts for their representativeness of the labor market to include low- and high-salary occupations and for being qualitatively important to BGT clients. We can see that for model15, the value at each percentile is lower in each case. So, for example, in the least 10% of absolute deviations, the base model has a maximum deviation of 562 and model15, 362. This is one indication that aggregate data is does not adversely effect MAE which needed to be tested to address Item (2) in the Introduction.



(a) Comparison of model15 to the base model outliers



(b) Comparison of the base model to model15 outliers

Figure 1: Model comparison of outlier predictions

Here each circle represents the percent change from the gold salary to the predicted salary for a single job posting. In (a), this shows how model15 generalizes on the base model's extreme outliers. The x-axis is the percent change for the base model which shows the 10th and 90th quantile outliers. The y-axis is percent change of the given posting for model15. Similarly, in (b), this shows how the base model generalizes on the model15's extreme outliers. The x-axis is the percent change for model15 and the y-axis is percent change from the gold salary for the base model. The regression line shows a line of best fit for the shown points. A coefficient of zero would indicate a perfect model for the y-axis. So, a smaller coefficient is better.

If instead of looking at the 80% of the data that falls within the 10th and 90th percentiles, we look at the most extreme 20% of deviations beyond these percentiles, then we can also see a compression of these predictions toward the gold salary. Figure 1 shows the *percent change* between the gold salary and the predicted salary. For example, in Figure 1a an individual circle represents the percent change from the gold salary for single job posting. The x-value of the circle is the percent change from the gold salary of the base model and the y-value is the percent change from the gold salary of model15 for the same posting. The postings represented in 1a are the 10th and 90th percentile most extreme outliers from the gold salary for the *base model*. The gap along the x-axis represents the postings which lie between the 10th and 90th percentiles. The y-axis shows how model15 behaves on the postings which deviated greatly from the gold salary with the base model. So, Figure 1a shows the generalization of model15 on the extreme predicted deviations from the base model and Figure 1b shows the generalization of the base model on the extreme predicted deviations from model15. If a circle lies within the two acute angles of the red lines then this indicates the predictions on the y-axis were no worse then the predictions on the x-axis. Therefore, if the model predictions for the y-axis are better then most of the points should fall within these two regions. The dashed red line is a softer criteria of boundedness because the *absolute value* of the percent change could still be less for the y-axis predictions if it falls outside this boundary, when compared to the x-axis value. The off-diagonal (orange) line is a linear regression fit for these points. Because the points should be centered in the acute regions, a coefficient closer to zero would indicate a model which behaved more accurately on these postings. The coefficient for the model comparison in Figure 1a is 0.70 and the coefficient for the model comparison in Figure 1b is 0.79. This indicates that model15 predictions generalized better in Figure 1a then how the base model behaved on the outlier predictions of Figure 1b.

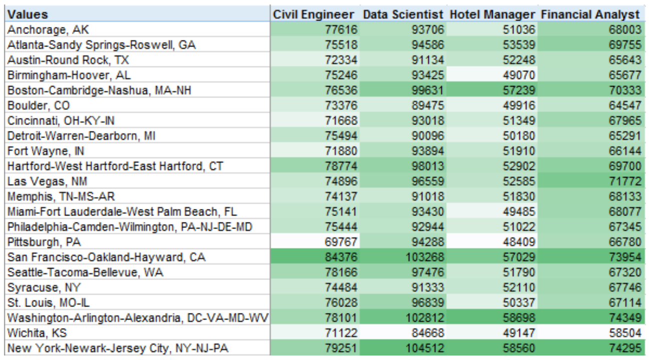
The histograms in Figure 1 shows the distribution of the points along the x- and y-axes respectively. In Figure 1a we can see that the distribution along the y-axis is clearly centered around zero. This shows that model15 was able to generalize on greatest errors of the base model. In Figure 1b the histogram along the y-axis is more bi-modal then that in Figure 1a. This indicates that while the base model was able to make more reasonable predictions on some of the outlier prediction of model15, it was not able to generalize as well to have a uni-modal distribution centered around zero of Figure 1a.

6.3 Comparison to External Sources

Being reliant on a single source of data can be problematic if there is extreme noise or consistent errors in the data. Having the ability to validate your findings against an external data source brings its own challenges, but it also provides an opportunity corroborate your findings. The Occupational Employment Statistics (OES) [1] annually publishes salary data for government classified occupations (SOC). The BGT occupational taxonomy is much more granular than SOC coding. In the BGT occupational hierarchy, BGTOcc is the most similar level of granularity to OES occupational codes. Because of differences in occupational coding, salary comparison is not completely precise. Table 6 shows the mean aggregates of gold, predicted, and OES salaries. We see that the aggregate salaries between gold and predicted salaries are similar. Comparison between predicted and OES are also reasonable. However, there are some BGTOccs where the difference between predicted and OES salaries are quite different. For example, see *Police Chief / Sergeant* in Table 6. We have found two reasons for these discrepancies: 1) as

Table 6: Salary Comparison of BGTOcc Aggregates to OES

	Mean Salary		
BGTOcc	Gold	Predicted	OES
Barista	23878	23912	19970
Business Intelligence Analyst	77310	77294	86805
Chief Information Officer	145384	142045	135624
Civil Engineer	78989	78459	83540
Computer Support Specialist	43236	43049	52025
Cyber Security Engineer	96662	95389	91960
Data Scientist	123331	121625	111840
Financial Analyst (General)	61593	63304	81665
General Manager	70255	71005	99310
Home Health Aide	29380	29378	22600
Nursing Home Administrator	77228	77142	96540
Police Chief / Sergeant	49973	51673	84840
Registered Nurse	68562	68649	68502



Values	Civil Engineer	Data Scientist	Hotel Manager	Financial Analyst
Anchorage, AK	77616	93706	51036	68003
Atlanta-Sandy Springs-Roswell, GA	75518	94586	53539	69755
Austin-Round Rock, TX	72334	91134	52248	65643
Birmingham-Hoover, AL	75246	93425	49070	65677
Boston-Cambridge-Nashua, MA-NH	76536	99631	57239	70333
Boulder, CO	73376	89475	49916	64547
Cincinnati, OH-KY-IN	71668	93018	51349	67965
Detroit-Warren-Dearborn, MI	75494	90096	50180	65291
Fort Wayne, IN	71880	93894	51910	66144
Hartford-West Hartford-East Hartford, CT	78774	98013	52902	69700
Las Vegas, NM	74896	96559	52585	71772
Memphis, TN-MS-AR	74137	91018	51830	68133
Miami-Fort Lauderdale-West Palm Beach, FL	75141	93430	49485	68077
Philadelphia-Camden-Wilmington, PA-NJ-DE-MD	75444	92944	51022	67345
Pittsburgh, PA	69767	94288	48409	66780
San Francisco-Oakland-Hayward, CA	84376	103268	57029	73954
Seattle-Tacoma-Bellevue, WA	78166	97476	51790	67320
Syracuse, NY	74484	91333	52110	67746
St. Louis, MO-IL	76028	96839	50337	67114
Washington-Arlington-Alexandria, DC-VA-MD-WV	78101	102812	58698	74349
Wichita, KS	71122	84668	49147	58504
New York-Newark-Jersey City, NY-NJ-PA	79251	104512	58560	74295

Figure 2: Mean Aggregate SubOcc Predictions by MSA

We see a general stratification of salary by MSA. For example, San Francisco, CA generally has the highest salaries and Wichita, KS the lowest.

we have mentioned, there are differences in occupational mapping and 2) we have found labeling issues in the gold dataset, which tend to be specific to certain occupations. Each individual discrepancy we have found has not been fully identified into which category it belongs or if the discrepancy is due to both items.

6.4 Qualitative Analysis and Semantic Learning

A very important aspect of this salary model is that it should confirm expectations in some reasonable way. This model is designed to learn subtle, but important, distinctions in the data to derive a semantic understanding of the individual features and is meant to address Item (3) in the Introduction. A specific example here would be location. The expectation is that a *Civil Engineer* would earn more in Seattle, WA then they would in a smaller, less expensive area such as Sandy Springs, GA. The location feature in the BGT dataset is MSA, of which there are 388. Each MSA (as well as all other features) *embeds* meaning. This semantic understanding of features is used to adjust salary in a meaningful way.

Table 7: Title Suffix Perturbation

Title	Pred. Salary	Pct. Change
Software Engineer I	82417	-
Software Engineer II	92154	11%
Software Engineer III	98326	19%
Software Engineer IV	100076	21%

Table 8: Education and Experience Perturbation

Title	Feature	Pred. Salary	Pct. Change
Legal Editor	High School	41135	-
Legal Editor	Associate	48748	18%
Legal Editor	Bachelor	50072	21%
Legal Editor	Masters	57169	38%
Legal Editor	PhD	54046	31%
Data Scientist	High School	87497	-
Data Scientist	Associate	94464	7%
Data Scientist	Bachelor	106992	22%
Data Scientist	Masters	107729	23%
Data Scientist	PhD	119983	37%
Quality Assurance	Exp: 2	67042	-
Quality Assurance	Exp: 4	71061	6%
Quality Assurance	Exp: 6	74881	11%
Quality Assurance	Exp: 8	76128	13%
Quality Assurance	Exp: 10	82201	22%

Figure 2 shows a table of mean aggregate salaries by MSA for a number of SubOcss from the hand-selected listed mentioned above. The thing to notice here is that salaries for a given MSA tend to be higher or lower across SubOccs. In other words, these predictions are relatively higher in higher paid locations and lower in lower paid locations. The model has learned that different location entities should effect the salary in different ways, even on data that may not have been observed during training.

For inputs into the model, title is the only *free text* field. Because of this lack of structure for this field it has the potential to hold valuable salary information. To test varying title information we devised a set of simulated job postings where we hold all features constant for a single test and perturb a single feature—in this case title—to see how the salary varies under these conditions. Some of these test were very subtle as shown in Table 7, where we change the suffix of the title (a one or two character difference) to see how the salary varies while changing the suffix to a title. We can see there is an increase in salary between these different levels of Software Engineer.

In addition to titles, we also looked at perturbing education and experience while holding the rest of the features constant. Table 8 show two rôles as their education and experience change and all other features remain constant. We see that for a Legal Editor with a PhD (an unlikely scenario to begin with) the predicted salary decreases slightly. This is a general phenomenon that this pattern holds for higher-level titles, more experience, or more education.

This pattern holds for the gold salary in the training set as well. It is currently unclear if this is due to a sparsity of training data at these levels or if this is a true reflection of the labor market. We do see, however, a general increase in salary as education and experience increase.

7 DROPPED FEATURE ANALYSIS

Neural network models are generally not conducive to interpretability. For this salary model, however, a way interpret the input features was needed to indicate their relative importance measured by their influence on the predicted salary. Here we introduce a novel method to interpret input features and their impact on the model output. This method is independent of the model itself and the benefits described in this section are optionally performed at prediction time.

Given a set of input features we can make a prediction with the model. If we drop a single feature prior to input, we may also obtain a prediction with the model. This prediction will be different, it may be greater than or less than the original prediction. If the predicted value is less than the base salary (all features are included) this indicates that this feature is important to this input set. The reason being is that if we insert this feature back into the feature set, then it will increase the predicted value. Similarly, if a removed feature increases the predicted salary, this feature has a negative influence. This process, described for a single feature, can be performed serially for the set of input features. We will now have a number of predictions; these predictions may be averaged to obtain a final ensembled prediction. This method is Dropped Feature Analysis (DFA). In Table 9 we see a base salary, the impact of each feature on the prediction, including the tokenized title, and a final averaged value. In Table 9 Delta defined by $delta = base\_salary - predicted\_salary$, and Delta Norm is defined by $delta\_norm = delta/base\_salary$. We can see here that location has the largest relative impact on the salary. For the averaged salary this is a form of auto-ensembling and is similar to [14] who uses a similar strategy for model testing. Their proposed Leave-One-Out training strategy is not employed here.

Most ensembling methods average predictions over a set of models. This methodology allows us take advantage of a variance-reducing ensemble without the overhead of utilizing multiple models. Using DFA predictions as an averaging ensemble is a variance-reducing technique. As shown in Table 3, model15 had an MAE of 3844. Utilizing DFA on model15 gives an MAE of 3553 as seen in Table 3, model 15a. This is an additional 7.5% reduction in MAE. While this method is more computationally expensive at prediction time, this provides a simple yet effective way of reducing additional variance in the model. For a particular set of input features one single feature may have a disproportionate impact on the predicted salary. This may stem from the feature itself (e.g. a high-value skill) or an anomalous result due to data sparsity. By averaging this ensemble, we constrain the impact of features with a large influence on the predicted salary.

Dropped Feature Analysis also allows us to have insight into the relative importance of a single input feature. As Breiman indicates in [15], feature importance (variable importance) methods provide insight at the global level and they indicate which features have the highest predictive value. DFA on the other hand operates at the record level. This provides a micro instead of a macro view of the importance its individual features. We would like to assess the importance of the input features, rather than assess the importance of the numeric input parameters. This would not be possible with variable importance because of the data pre-processing; hashing of the features obscures the identity of numeric input parameters. Variable importance is used to find features with the highest predictive value. DFA discerns feature importance through dropping individual features and finding a feature's relative impact when compared to the base salary. Further, this ensemble is used to improve predictive performance. To recapitulate, DFA has a number of facets of interest when compared to Breiman's variable importance:

- a single model is used instead of a forest of models,
- feature importance is detailed at the record level instead of a global level,
- instead of discerning feature importance by its predictive performance, DFA *improves* predictive performance,
- DFA is a variance-reducing ensemble method, and
- it is possible to assess the importance of raw text features as opposed to just the numeric model input parameters. This would not be possible with Breiman's method due to data pre-processing.

Table 9: Dropped Feature Analysis: Enterprise Software Engineer Because title is tokenized the title features of this Dropped Feature Analysis are prefixed below with "title:" for clarity. Here Delta is the difference of the predicted salary from the base salary. Delta Norm is the normalized Delta relative to the base salary.

Dropped Feature	Delta	Delta Norm
base	97441	-
San Francisco-Oakland, CA	14295	15%
Experience: 5 yr.	9817	10%
title: Engineer	7331	8%
Ajax	7209	7%
Building Effective Relationships	4957	5%
Apache Subversion	4846	5%
Extensible Markup Language	4220	4%
Communication Skills	3078	3%
title: Enterprise	1954	2%
Enterprise Java Beans	1215	1%
title: Software	-565	-1%
Eclipse	-1367	-1%
Hardware/Software Installation	-3548	-4%
Customer Service	-3891	-4%
averaged salary	94137	-

8 CONCLUSION AND FUTURE WORK

In this paper we looked at the Burning Glass data and taxonomy and how this data is transformed for use in a deep neural network model. This feature space to parameter space transformation utilizes input feature hashing to reduce the dimensionality of the network input and parameter weights use a modified TF-IDF weighting (TF-ICF) relative to an individual posting's OccGroup. The TF-ICF weighting for the base model reduced MAE model error by 8%. The final selected model, model15, reduced the MAE another 9% to 3844, when compared to the base model. Leveraging the proposed Dropped Feature Averaging, we reduced the MAE to 3553, a 16% reduction from the base model. It is not an exact comparison because of the difference in the features of the models, but when comparing the MAE of the Adzuna competition mentioned in the introduction the winning MAE was 5403 (adjusted to USD) while the best performing BGT salary model obtained an MAE of 3553. Here we leveraged DFA as a variance-reducing ensemble method and we also pursued how this method may provide insight into record-level feature importance.

In addition to looking at MAE for model selection and variance reduction, we also looked at the qualitative aspects of this model and its learning of semantic structure. Model input can be very nuanced and these small changes in input can set up expectations for how the model should behave. We saw this, e.g. in title suffix changes to indicate different levels of a rôle.

After production deployment of this model, we will populate existing postings with predicted salaries. There will also be a salary API which will handle real-time, ad-hoc salary requests. These requests will be logged so that we may evaluate the performance of the model in production relative to aggregate statistics of the gold salary data. Additionally, we will work closely with clients to obtain first-hand knowledge of their experience and interaction with the product.

To continue future research, we want to look at DFA and its ability to generalize to classification tasks. This method is agnostic to model type in this regard, as long as the output of the classification is a probability distribution. So this could be used for ensemble averaging as well as feature importance based on percent change from the base class probability. We also want to further investigate additional external data sources to corroborate our data findings. In particular, we could investigate the salary changes based on location and how these deviations compare to cost of living ratios between cities. Lastly, based on discrepancies found in our comparison with OES data, we want to further investigate if large deviations from OES salary data indicates a misalignment with the OES occupation to BGTOcc mapping or if BGT salary tagging is the source these deviations. Similarly, we want understand the depression of salary we see for postings with many years of required experience or high educational requirements, to see if this is related to training data sparsity, data tagging, or if it is a true reflection of the labor market.

ACKNOWLEDGMENTS

We would like to that all of those involved in the development of this model and for determining and understanding all of the possible applications of this model. In particular, we would like to thank Dan Restuccia and Christopher Dedels for their invaluable insight and understanding of the complexities of this modeling task. We would also like to thank the BGT Applied Research team for their contributions and wholistic view of this process.

REFERENCES

[1] U.S. Department of Labor Bureau of Labor Statistics. 2017. Occupational Employment Statistics. (2017). http://www.bls.gov/oes/
[2] LLC. CareerBuilder. 2016. How to Rethink the Candidate Experience and Make Better Hires. *CareerBuilder's Candidate Behavior Study* (2016).
[3] François Chollet et al. 2015. Keras. https://github.com/keras-team/keras. (2015).
[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385
[5] Kaggle Inc. 2013. Job Salary Prediction | Kaggle. (2013). https://www.kaggle.com/c/job-salary-prediction
[6] Stack Exchange Inc. 2017. 2017 Average Software Developer Salary - Stack Overflow. (2017). https://stackoverflow.com/jobs/salary
[7] XE.com Inc. 2018. XE: GBP / USD Currency Chart. British Pound to US Dollar Rates. (2018). http://www.xe.com/currencycharts/?from=GBP&to=USD&view=10Y
[8] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* abs/1502.03167 (2015). arXiv:1502.03167 http://arxiv.org/abs/1502.03167
[9] Krishnaram Kenthapadi, Ahsan Chudhary, and Stuart Ambler. 2017. LinkedIn Salary: A System for Secure Collection and Presentation of Structured Compensation Insights to Job Seekers. *CoRR* abs/1705.06976 (2017). arXiv:1705.06976 http://arxiv.org/abs/1705.06976
[10] John Langford. 2007. Vowpal Wabbit Code Release. (2007). http://hunch.net/?p=309
[11] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad Click Prediction: a View from the Trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).*
[12] Tom Mitchell. 1997. *Machine Learning* (1st. ed.). University of Chicago Press, Chicago. https://doi.org/10.1007/3-540-09237-4
[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[14] R. Sabourin, P. R. Cavalin, A. de Souza Britto, and A. H. Ko. 2008. Leave-One-Out-Training and Leave-One-Out-Testing Hidden Markov Models for a Handwritten Numeral Recognizer: The Implications of a Single Classifier and Multiple Classifications. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 31 (10 2008), 2168–2178. https://doi.org/10.1109/TPAMI.2008.254
[15] Leo Breiman Statistics and Leo Breiman. 2001. Random Forests. In *Machine Learning*. 5–32.
[16] Deqing Wang, Hui Zhang, Wenjun Wu, and Mengxiang Lin. 2010. Inverse Category Frequency based supervised term weighting scheme for text categorization. *CoRR* abs/1012.2609 (2010). arXiv:1012.2609 http://arxiv.org/abs/1012.2609