are the contexts of the tri-letters, as well as the position of the tri-letters. BERT is not applied in this task, because the typeahead product has a strict latency constraint. The latency would not meet the production constraint for BERT is too computationally heavy.

4.3.1 Multilingual Support. Search systems often experience a large amount of traffic in many countries with different languages. In LinkedIn typeahead search, more than 30% of search traffic is from languages other than English. Handling the performance gap between English and other languages is crucial to enable a good user experience in typeahead.

The typical way to handle multilinguality is to train per-language models, which is not easy to maintain and scale. The character-level approach can be easily extended to support multiple languages with a unified model, since the character vocabulary of all languages is much smaller than the word vocabulary.

To train a unified multilingual model, two approaches are designed to model the language/locale information:

Language Feature Embedding: The first approach is to concatenate the interface locale embedding to the input query embedding, and then feed it through a BiLSTM to generate interface locale âĂIJembeddedâĂİ query features, as shown in Figure 3a. The embeddings of the interface locales are randomly initialized and together trained with the network.

Language Feature Concatenation: The second approach is to use the BiLSTM layer for extracting query features from query embeddings, and then concatenate the interface locale embeddings to the last output state of the BiLSTM, as in Figure 3b. Compared to the first approach, BiLSTM is unaware of the language.

4.4 Complete Query Intent Modeling

We design word-level CNN, LSTM, and BERT based models for complete query intent classification. CNN is good at capturing word n-gram patterns. LSTM is powerful for modeling long sequence context. BERT further enriches the query representation by modeling the context meaning. Specifically, a pre-trained BERT model with LinkedIn data (**LiBERT**) is used.

There are two major motivations on pre-training a BERT model with in-domain LinkedIn data (**LiBERT**) instead of using the off-the-shelf BERT models released by Google [9]:

Latency Constraint. The probabilities of different intents given a query need to be computed on-the-fly instead of pre-computing as there are unique queries from different users searched each day. Given that GPU/TPU serving is not yet available at LinkedIn, the latency for computing the query intent outputs using the BERT-Base model from Google wonâĂŹt satisfy the critical latency requirement. The LiBERT model has a lighter architecture (discussed in Section 5.3.2) than the smallest model released from the original paper [9].

Better Relevance Performance. In addition to the benefit of low latency, pre-training the BERT model from scratch with LinkedIn data leverages the in-domain knowledge in the corpus, therefore can provide better relevance performance for downstream fine-tuning tasks. An example is that `LinkedIn`, `Pinterest` are both out-of-vocabulary words in GoogleâĂŹs BERT-Base model. These company names are generally important for LinkedIn search.

Note that we haven't explored a multilingual approach for the word-level complete query intent models. This is because the word-level vocabulary could be fairly large if the major languages are included and therefore adding more complexity to the models.

5 EXPERIMENTS

Experiments on both incomplete and complete query intent prediction models are discussed in this section. Performance comparison in both offline and online metrics are shown by relative differences instead of absolute values due to company policy. We also share our analysis in the scalability of these deep learning based models, as well as the online A/B testing performance of these models compared with production baselines. Tensorflow serving with CPU is used for online inference for the deep models. Traditional features such as the user behavioral features are pushed to a online store after daily offline computation and can be accessed during model inferencing. All the reported online metrics lift are statistically significant ($p < 0.05$) and are collected based on 4 weeks of 50% A/B testing search traffic. The LSTM model and BERT model are fully ramped to production after the A/B testing for incomplete query intent and complete query intent, respectively.

5.1 Training Data Collection

Search click-through log is used for training data collection. For example, given that a user searched for *linkedin sales solutions* and clicked on a job posting instead of a *people* or *content* page, a *job* intent label is assigned to this query. Similarly, when a user typed an incomplete query *face* and chose the *facebook* company page, then a *company* intent is inferred.

For both incomplete and complete intent models, we collected training data sampled from clicked-through log for a month. For train/dev/test dataset, we sampled 42M/21K/21K for English incomplete models, 63M/36K/36K for multilingual incomplete query models, and 24M/50k/50k for complete query intent models.

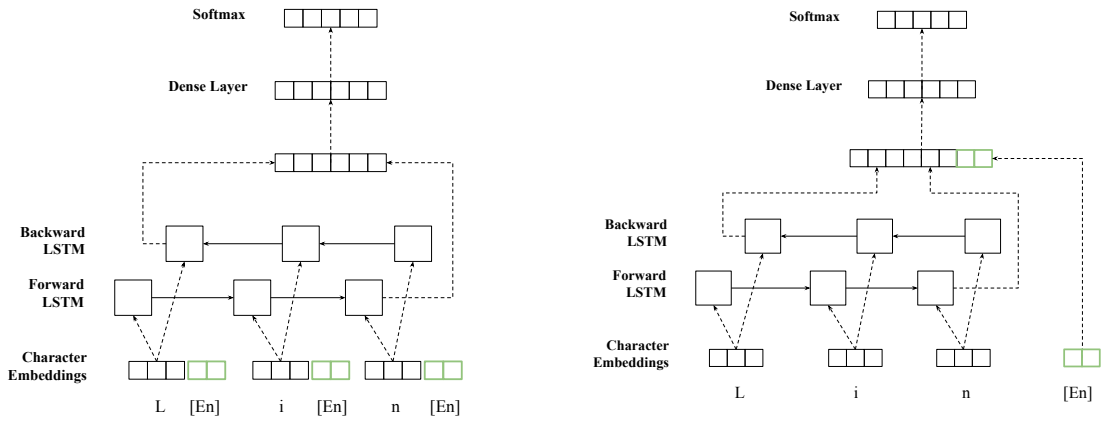
5.2 Incomplete Query Intent Prediction

5.2.1 Experiment Setting. We evaluate the performance for character-based CNN and LSTM models in the offline experiments. The baseline model is learnt using letter tri-gram bag-of-words features with a logistic regression classifier.

In the CNN/LSTM based models, character embeddings of size 128, with a 500 vocabulary size are used. The characters in the vocabulary are extracted from the most frequent characters in the training data. The embeddings are randomly initialized at the beginning of training. We used 128 convolutional filters of size 3 for training the CNN model. In the Bidirectional LSTM model, the same character vocabulary is used as in the CNN model and the number of hidden units is 128.

5.2.2 Offline Experiments. Both English and multilingual incomplete query models are investigated in offline experiments. Throughout experiment result comparison, we use the $-en$ and $-i18n$ notation to represent English models and multilingual models.

English Model. The first set of experiments were conducted on English incomplete queries. Experimental results show that compared with the tri-letter model, the CNN-based model CNN-en is



(a) Embedding. **(b) Concatenation.**

Figure 3: The language feature embedding and concatenation scheme for multilingual character-level query intent model.

Table 1: Offline performance comparison for different character-level models vs production baseline.

Model	Accuracy	F1 (people)	F1 (company)
Tri-letter	-	-	-
CNN-en	+4.14%	+2.69%	+11.39%
LSTM-en	+7.11%	+4.46%	+20.36%

Table 2: The size of the character vocabulary for incomplete query intent models.

Model	Vocab	Accuracy	F1 (*people*)	F1 (*company*)
Tri-letter	-	-	-	-
CNN-en	70	+4.09%	+2.69%	+11.38%
CNN-en	500	+4.14%	+2.73%	+11.75%
LSTM-en	70	+7.04%	+4.46%	+20.36%
LSTM-en	500	+7.11%	+4.49%	+20.49%

able to achieve a higher accuracy, which is mostly because of the fact that CNN could extract more abstract word pattern features than traditional bag-of-words features. As shown in Table 1, the Bidirectional LSTM model LSTM-en can further improve the performance (+7.11% vs. +4.14%), since LSTM can better model the long range character sequences.

We also compare the impact of character vocabulary size for the deep models (Table 2). It is interesting to see that even with a small vocabulary (only 70 characters consisting of lower-case English characters and special characters), the accuracy is significantly higher compared with the baseline model.

Multilingual Model. In the multilingual experiments, we first collect training data using a similar procedure on top 30 international locales, such as French, Portuguese, German, etc. The international locales are used as language feature in our multilingual model. The most frequent 10k characters are extracted as vocabulary.

Table 3: Multilingual models for character-level incomplete query intent.

Model	Accuracy	F1 (*people*)	F1 (*company*)
LSTM-i18n	-	-	-
LSTM-i18n-embed	+0.67%	+0.44%	+1.34%
LSTM-i18n-concat	+0.55%	+0.37%	+1.12%

Table 3 summarizes the performance of two different strategies discussed in Section 4.3.1. The results suggest that both methods outperform the baseline without language embedding injected. The language feature embedding method (LSTM-i18n-embed) outperforms the concatenation (LSTM-i18n-concat) method slightly. This might be that in LSTM-i18n-concat, the LSTM is aware of the language, hence generating a more meaningful representation for the character input sequence.

We further compare the effectiveness of the multilingual models to the traditional per-language models. Table4 shows the overall accuracy and per-language accuracy. The LSTM-per-lang is trained and evaluate on each individual language portion. LSTM-i18n is the multilingual model without language features added. LSTM-i18n-embed is with language feature added to the character embeddings. The overall accuracy for LSTM-i18n-embed is comparable to the per-language model. We also sampled 4 languages to show the individual performances. For French (fr) and Chinese (zh), both multilingual models improved performance compared with the single language LSTM-i18n-per-lang model trained on data from single language only. This is true that for certain languages, multilingual helps the model to learn from other languages and generalize better to help predictions on the specific languages. From results in the other 2 languages, both LSTM-i18n and LSTM-i18n-embed did not outperform the per-language models. This is often expected. But with the language feature embedded (LSTM-i18n-embed), we observed an increased accuracy compared to the LSTM-i18n model.

Table 4: Multilingual model vs per-language models.

Model	Overall	fr	de	zh	da
LSTM-i18n-per-lang	-	-	-	-	-
LSTM-i18n	−0.65%	+0.28%	−0.84%	+0.54%	−1.24%
LSTM-i18n-embed	+0.02%	+0.56%	+1.01%	−1.11%	−0.66%

Table 5: Online performance comparison for incomplete query intent models vs baseline linear model.

Model	Traffic	Metrics	Lift
LSTM-en	English	Search session success	+0.43%
		Time to success (mobile)	-0.15%
LSTM-i18n-embed	Non-English	Search session success	+0.86%
		Time to success (mobile)	-0.19%

Table 6: Model architecture comparison between LiBERT Google's BERT-Base.

BERT HParams	BERT-Base	LiBERT
Layers	12	3
Hidden size	768	256
Attention heads	12	8
Total #params	110M	10M

5.2.3 Online Experiments. As shown in Table 5, the English model (LSTM-en) and multilingual model (LSTM-i18n-embed) are tested in typeahead search production with English/non-English traffic, respectively. The online business metrics measure the success criteria within search sessions. Both models show significant performance gains for *search session success*. An interesting observation is that the *average time to success* on mobile is reduced for both models. An intuitive reason is that mobile users are more likely to be click on the high quality results rendered in typeahead search due to the typing behavior compared with on desktop.

5.3 Complete Query Intent Prediction

5.3.1 Experiment Setting. The input is word-level embeddings. The embeddings are initialized using GloVe[22] word vector representations pre-trained on LinkedIn text data. It covers a vocabulary of 100K with dimension 64.

The production baseline model is a logistic regression model, with bag-of-words features, and user profile/behavioral features, etc. In the offline experiments, we first compare the performance among the deep learning models with the production baseline. For the CNN-based model, we used 128 filters of height 3 (tri-gramd) for the 1-D convolution. The hidden state size is 128 for the LSTM-based model. For both CNN and LSTMs, after the query embedding is generated, a layer of size 200 is used to capture the non-linear interactions between query representation and traditional features.

5.3.2 LiBERT Pre-training. The LiBERT-baesed model is fine-tuned with a BERT model pre-trained on LinkedIn data. The pre-training data include a wide variety of data sources across LinkedIn: search queries, user profiles, job posts, etc. The collected data for pre-training include around 1 billion words. A light-weight structure is used compared to the BERT-Base (a smaller model published in BERT [9]). A comparison between BERT-Base and LiBERT is given in Table 6.

Table 7: Offline performance comparison for the word-level models vs production baseline.

Model	Accuracy	F1 (*people*)	F1 (*job*)
LR-BOW	-	-	-
CNN-word	+6.40%	+1.36%	+17.35%
CNN-char	+3.49%	−0.50%	+10.29%
LSTM-word	+6.69%	+1.47%	+17.99%
LSTM-char	+4.63%	+0.23%	+13.46%
BERT-Base	+8.20%	+2.33%	+19.33%
LiBERT	+8.35%	+2.46%	+20.60%

5.3.3 Offline Experiments. Table 7 summarizes the performance of different models. CNN/LSTM outperforms the baseline models, by +6.40%/+6.69%, respectively.

We further compared the performances of the complete query models in different token granularity and deep model types. First, the word-level models outperforms character-level models consistently. This implies that for complete queries, word level representations captures more meaningful features than character sequences. Next, we compare the performance in LSTM and CNN models under different token granularity. More specifically, on word-level models, LSTM and CNN have similar performances. This implies that the useful sequential information is limited in the word sequences due to the short length of the queries and possibly the effectiveness of word-level information such as word meaning. This is further proved as when comparing the character-level models, LSTM outperforms CNN by a larger margin, meaning that longer range information is more useful in character sequence modeling. Similar observation can be found in Table 1 on incomplete query intent model experiments, where we saw even more significant improvement on LSTM over CNN, implying that the LSTM is more suitable in modeling the incomplete character sequences in typeahead than the complete queries in SERP.

BERT models (BERT-Base and LiBERT) can further improve the accuracy over CNN/LSTM models. This can be attributed to the contextual embeddings that are able to disambiguate words under different contexts better.

In addition, we'd like to analyze the effect of traditional features on the deep models. In As shown in Table 8, without traditional features, the deep models (CNN, LSTM, and LiBERT) bring much less performance gain compared with those that incorporate these features in a wide-and-deep fashion.

5.3.4 Online Experiments. It is worth noting that even though the LSTM model performs slightly better than the CNN model, we proceed to implement the CNN model in online production. This is due to the fact that LSTM does not bring much relevance gain while introducing almost 2 times inference latency (Table 10).