A Neural Language Model for Query Auto-Completion

Ссылка: https://www.researchgate.net/publication/317379370_A_Neural_Language_Model_for_Query_Auto-Completion#pf4

Год: 2017

Решают задачу «query auto-completion (QAC)».
Архитектура: 2 слоя LSTM на character-level эмбеддингах. Немного улучшают метрику добавлением к эмбеддингу символа эмбеддинг слова(только для пробела, для остальных - это «<INC>»).
Лучше всего метод работает, когда дополняет MPC (Most Popular Completion). Главная фишка: в отличие от MPC, это решение **может предсказывать ввод, который никогда не видело**.
Для экспериментов берут предобученные на Google News дотасете эмбеддинги и обучают на своих (yahoo) данных. За счет unseen данных улучшают MRR MPC почти в 2 раза. Про **время работы** скромно умалчивают.

Personalized Language Model for Query Auto-Completion

Ссылка: https://arxiv.org/abs/1804.09661

Год: 2018

Опираются на предыдущую статью, хотят сделать **personalized-QAC**, для этого пробуют 2 разных способа **добавить эмбеддинги юзеров** в модель: в первом случае, просто добавляют к эмбеддингу символа эмбеддинг юзера; во втором, добавляют к матрице весов матрицу, которая хитро зависит от эмбеддинга юзера (в некотором роде, у каждого юзера своя модель). Обучаются и тестируются на **AOL Query data**. Сильно лучше первой статьи не становится.

Personalized neural language models for real-world query auto completion

Ссылка: https://arxiv.org/abs/1804.06439

Год: 2018

Снова **опираются на первую статью**, дополняют эмбеддинг символа **эмбеддингом юзера** и **sin и cos от времени**. И меняют LSTM на GRU.
Тут тоже обучаются и тестируют на **AOL**. MRR больше примерно на 10% по сравнению с 1-й статьей. Наконец-то сравнивают по времени: работает **в 4 раза дольше, чем MPC**, но зато в 2 раза быстрее, чем модель из первой статьи.

A Hierarchical Recurrent Encoder-Decoder For Generative Context-Aware Query Suggestion

Ссылка: https://arxiv.org/abs/1507.02221

Год: 2015

Решают задачу **предсказания следующего запроса в сессии.**
Архитектура: 3 GRU, первый-энкодер - получает энкодинг запроса. Второй - по энкодингам прошлых запросов генерирует энкодинг для следующего. Третий - декодер, из энкодинга предыдущей GRU генерирует запрос.
Обучаются и тестируют AOL. **Бьют бейзлайн ADJ** (не нашел, что это) **совсем немного**. Но **по некоторым срезам лучше**, например, для long-tail запросов, или medium/long sessions. В целом, кажется, это хорошое применение RNN для саджеста, по времени **decoding занимает ~1 секунду**, если верить статье.

Attention-based Hierarchical Neural Query Suggestion

Ссылка: https://arxiv.org/abs/1805.02816

Год: 2018

Продолжают предыдущую статью, **добавляя 4-й RNN на уровне юзера**, он работает с энкодингами RNN сессии. Второй версией **добавляют attention** на выбор user-RNN состояния session-RNN. По итогу, MRR версии с attention - 0.8514, MRR ADJ - 0.7072.

A Survey of Query Auto Completion in Information Retrieval

Ссылка: https://dl.acm.org/citation.cfm?id=3099948

Книга по теме, еще не читал, но хочу полистать.

Вывод: Мне кажется, 4-я и 5-я статьи можно применять в жизни, потому что время между соседними запросами в сессии существенно больше времени между вводом символов в одном запросе, поэтому RNN будет успевать подсказывать следующий запрос. Из-за чего я бы хотел заниматься задачами из 4-ой и 5-ой статей.