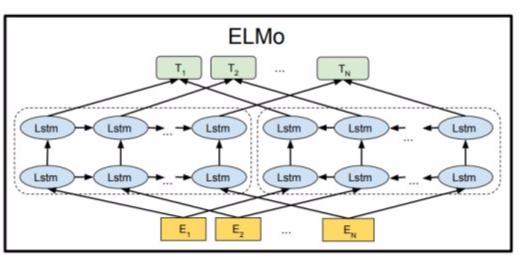
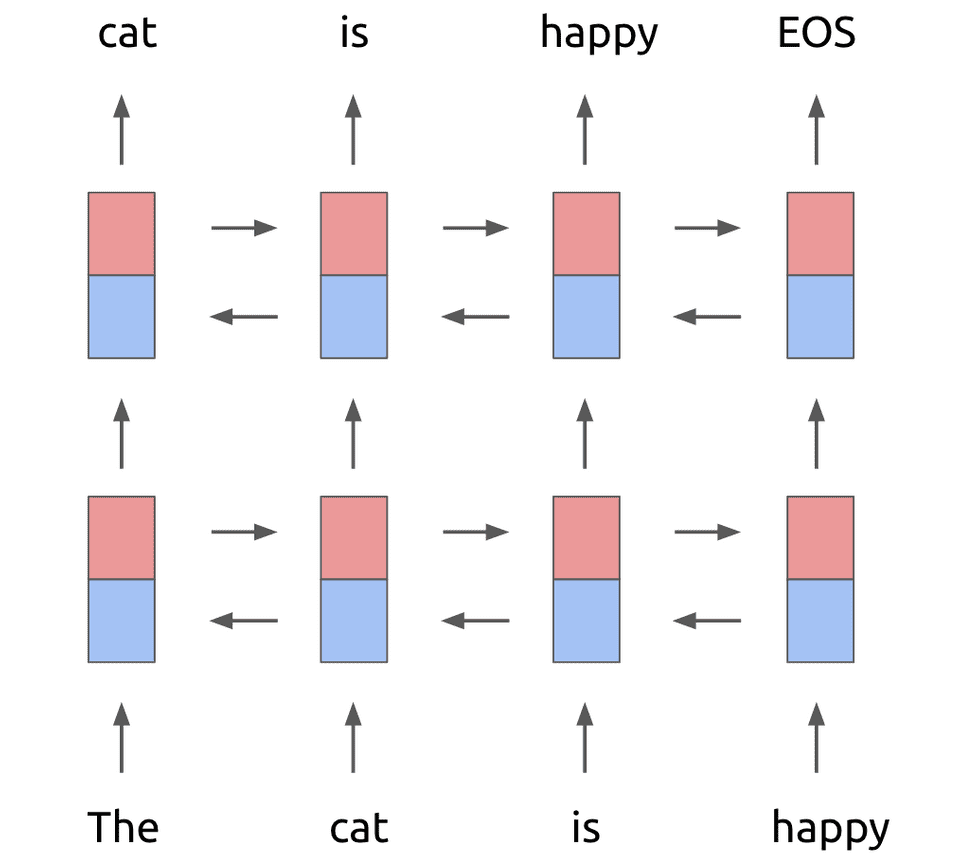
## ELMo

ELMo – это н**овое представление слов с глубоким контекстом**, Он может моделировать сложные особенности слов (такие как синтаксис и семантика) и изменения слов в языковом контексте (т.е. моделировать многозначные слова)

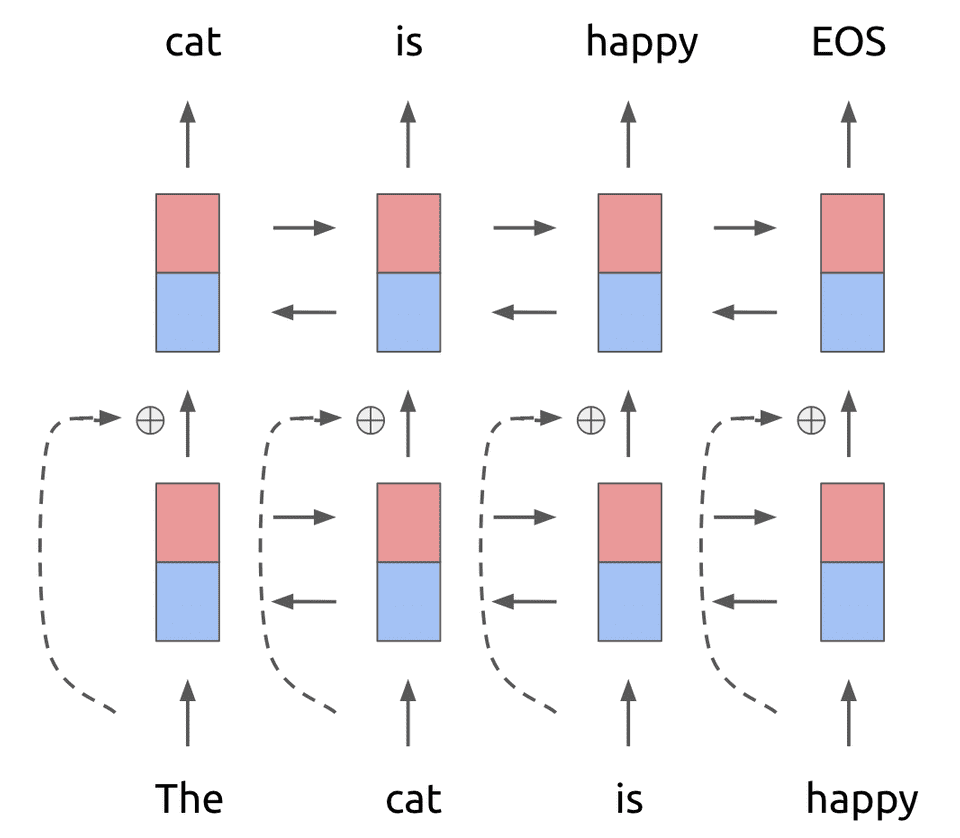
Как упоминалось ранее, наиболее важным моментом в ELMo является обучение языковой модели. Структура языковой модели выглядит следующим образом:



Для целей ELMo используемая языковая модель начинается с 2-слойной двунаправленной магистрали LSTM следующим образом:



Теперь к этой 2-уровневой сети добавляется остаточное соединение между первым и вторым уровнями. Высокоуровневая интуиция заключается в том, что остаточные связи помогают глубоким моделям тренироваться более успешно. Затем языковая модель выглядит следующим образом:



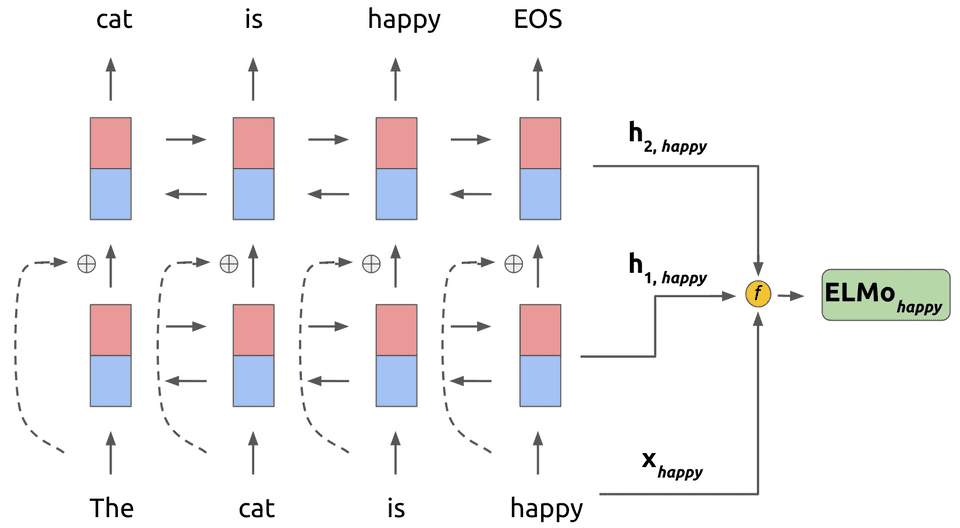
Теперь, в традиционных моделях нейронного языка, каждый токен в первом входном слое (в данном случае кот счастлив) преобразуется во встраивание слова фиксированной длины перед передачей в рекуррентную единицу. Это делается либо путем инициализации матрицы встраивания слов размером (размер словаря) x (размер встраивания слов), либо с помощью предварительно обученного встраивания, такого как GLoVe, для каждого токена.

Однако для языковой модели ELMo мы делаем что-то немного более сложное. Вместо того, чтобы просто искать встраивание в матрицу встраивания слов, мы сначала преобразуем каждый токен в соответствующее представление, используя встраивания символов. Это представление встраивания символов затем прогоняется через сверточный слой с использованием некоторого количества фильтров, за которым следует слой с максимальным пулом. Наконец, это представление передается через 2-слойную сеть магистралей, прежде чем быть предоставленным в качестве входных данных для уровня LSTM.

Эти преобразования во входной токен имеют ряд преимуществ. Во-первых, использование встраивания символов позволяет нам улавливать морфологические особенности, которые могут отсутствовать при встраивании на уровне слов. Кроме того, использование встраивания символов гарантирует, что мы сможем сформировать правильное представление даже для слов, не входящих в словарный запас, что является огромным преимуществом.

Далее, использование сверточных фильтров позволяет нам улавливать n-граммовые функции, которые создают более мощные представления.

Предположим, что мы смотрим на *k*атое слово в наших входных данных. Используя нашу обученную 2-слойную языковую модель, мы берем словесное представление *xk,* а также двунаправленные представления скрытого слоя и объедините их в новое взвешенное представление задачи. Это выглядит следующим образом:



*ELMoktask*​ = *γk*​⋅(*s*0*task*​⋅*xk*​+*s*1*task*​⋅*h*1,*k*​+*s*2*task*​⋅*h*2,*k*​)

## RoBERTa

Роберта выступает за надежный оптимизированный подход к предварительной тренировке BERT. Он был представлен исследователями из Facebook и Вашингтонского университета.

Изменения в BERT:

RoBERTa имеет почти такую же архитектуру по сравнению с BERT, но для того, чтобы улучшить результаты по архитектуре BERT, авторы внесли некоторые простые изменения в ее архитектуру и процедуру обучения. Эти изменения заключаются в:

* Удаление прогноза следующего предложения (NSP) цель: При прогнозировании следующего предложения модель обучается предсказывать, происходят ли наблюдаемые сегменты документа из одних и тех же или разных документов с помощью потери вспомогательного прогноза следующего предложения (NSP). Авторы экспериментировали с удалением / добавлением потерь NSP в разные версии и пришли к выводу, что удаление потерь NSP соответствует или немного улучшает производительность последующей задачи
* Обучение с большими размерами пакетов и более длинными последовательностями: Первоначально BERT обучается шагам в 1 м с размером пакета в 256 последовательностей. В этой статье авторы обучили модель с 125 шагами из 2 тыс. последовательностей и 31 тыс. шагов с 8 тыс. последовательностей размера пакета. Это имеет два преимущества: большие партии улучшают сложность задачи моделирования на замаскированном языке, а также точность конечной задачи. Большие пакеты также легче распараллеливать с помощью распределенного параллельного обучения.
* Динамическое изменение шаблона маскирования: В архитектуре BERT маскирование выполняется один раз во время предварительной обработки данных, в результате чего получается одна статическая маска. Чтобы избежать использования одной статической маски, обучающие данные дублируются и маскируются 10 раз, каждый раз с использованием другой стратегии маски в течение 40 эпох, таким образом, получается 4 эпохи с одной и той же маской. Эта стратегия сравнивается с динамической маскировкой, в которой различная маскировка для каждого раза, когда мы передаем данные в модель.