주요 ML problem 중 하나는 text classification이고, 그것의 예로는 spam을 구별하고, 뉴스 기사의 topic을 정의해주거나 다의어의 올바른 의미를 선택해 주기도 함

Statsbot 팀은 “how to train your own model for detecting spam emails, spam messages, and spam user comments.”을 썼었음

이번 기사에서는 우리는 Roman Trusov라는 데이터 과학자에게 더 자세한text 분석 ML을 요청함

너는 아마도 최적의 text classification이 무엇인지 정의하기 어렵다는 것을 알고 있을지도 모른다. Computer vision 같은 분야에서는 모델을 디자인 하는 일반적인 방법에 관한 아주 강력한 합의가 있다. 🡪 deep networks with lots of residual connections (수 많은 잔여 연결이 많은 deep network). 이와 달리, text classification은 좁은 영역 안에서도 합의가 이루어 지지 않고 있다.

이 기사에서 우리는 **text classification algorithm과 사용법 사례에 관한 몇 개의 주요한 일반화 된 접근**에 초점을 맞출 것이다. 높은 수준의 토론과 함께 우리는 tutorial 모음집과 자신의 모델을 설계할 수 있도록 도와주는 tool을 제공한다.

**Text Classification Benchmark**

TF-IDF 기능과 Linear SVM, word2vec과 같은 word embedding 등의text mining 범주와 주의기반(attention-based) neural 구조에 초점을 맞춘 현대 ML 종사자의 도구 상자

**특정 방법의 효과가 입증 될 때, 연구와 경쟁의 두가지 케이스를 구분 짓는 것은 매우 중요하다.**

연구자가 text classification algorithms을 비교할 때, 그들은 algorithm들을 동일한 문제에 대한 다른 많은 시도와 결과를 비교할 수 있는 잘 알려진 데이터 세트에 대해 몇 가지 트릭을 사용하여 그대로 사용한다. (뭔 개소리)

**Some well\_known text classification benchmarks**

* AG의 뉴스 기사
* Sogou 뉴스 코퍼스 (특정한 목적을 가지고 언어의 표본을 추출한 집합)
* 아마존 리뷰 전체
* 아마존 리뷰 양극
* DBPedia
* 야후 answers (지식인 같은)
* 옐프 리부 전체
* 옐프 리뷰 양극

우리는 **google drive에 special 폴더** 만들어 놨음 너 거기서 다운받아서 쓰셈

**Deep vs Shallow Learning**

NLP연구에 널리 채택된 데이터셋에 관한 정말 주목할 만한 점은 간단한 모델과 복잡한 모델 둘 다 잘 작동하는 것이다. 증명하기 위해 두 개의 논문에 대해 논의해 보겠음

* **A Bag of Tricks for Efficient Text Classification by Joulin et al**
* **Character-level Convolutional Networks for Text Classification by Zhang et al**

위 두개의 논문의 dataset은 같고, precision 측정에서도 두 실험의 결과가 대략적으로 같다. 그러나 두 모델 사이에 training과 추론 시간에는 굉장히 큰 차이가 있다.

처음 모델은 말 그대로 초 단위 훈련을 하는 반면, 두번째 모델은 몇 시간이 걸렸고, hyperparameter가 선택되면 그것은 게임 changer가 된다(하이퍼 파라미터에 따라 학습이 바뀐다는 뜻 같음)

이 접근을 흥미롭게 만드는 것은 그들의 모델은 데이터에 관해서 어떠한 가정도 만들이 않는다는 것이다. 최하위 수준에서 그들은 text를 characters의 연속으로 다루고 있으므로, convolutional layer가 완전히 내용을 인식하지 못하는 방식으로 기능을 구현하고 있다.

두번째 논문은 CPU상에서 빠르게 작동하도록 디자인된 훨씬 가벼운 모델을 특징으로 embedding layer와 softmax classification의 결합으로 구성된다.

**한편, 만약 네가 몇몇의 kaggle의 winning solution을 살펴본다면, 고도의 custom된 복잡한 ensemble에 의해 지배되고 있는 것을 볼 수 있을 것이다.**

가장 좋은 사례는 최근의 Quora Question Pairs competition과 현재진행중인 DeepHack.Turing 이다. Top rank의 솔루션은 Gradient boosting machine과 RNN, CNN 같은 몇 개의 다른 모델들로 구성된다.

여기서 우리가 배울 수 있는 **실용적인 교훈**은 연구에서 발표된 특정 방법의 결과임에도 불구하고 vivo 내의 특정 task에서 최고의 performance를 얻어내는 것은 괴학이기보단 art에 가깝고, 복잡한 파이프라인을 섬세하게 튜닝하는 것이 필요하다는 것이다.

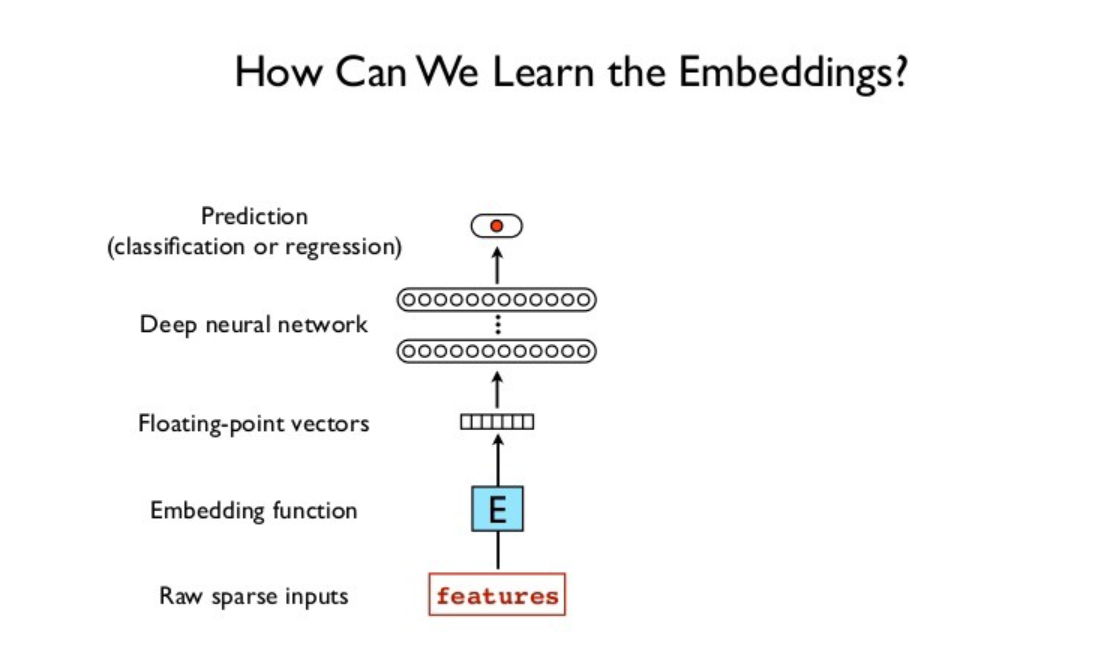
이 연구와 눈에 띄는 대조점은 Kaggle의 winning solution에 대한 기사에서 볼 수 있다. (뭔 소리?)

**Neural network-based text classifiers typically follow the same linear meta architecture (뉴럴넷 기반의 텍스트 분류기는 전형적으로 linear meta구조를 따른다)**

* Embedding
* Deep representation
* Fully connected part

**Embedding**

Embedding layer들은 input으로 단어의 index의 연속(단어의 matrix)으로 받고, vector간의 관계의 연속을 output으로 생성한다. 그들의 기능성은 매우 직관적이고, output vector의 실제 의미가 우리의 문제에서는 그닥 흥미로운 부분이 아니기 때문에, 우리는 오로지 “weights를 초기화하는 가장 좋은 방법은 무엇일까?”하는 질문만 계속 반복한다.



문제에 따라 “자신의 합성 label을 생성하고, word2vec을 train하고, 그것들의 embedding layer를 초기화하십시오”라는 조언과 직관에 어긋난다. (??)

그러나 모든 실제적인 목적을 위해 너는 미리 학습된 dataset의 embedding을 사용할 수 있고, 특정 모델에 맞게 미세하게 조정할 수 있다. 결과 word vector가 바닐라 word2vec 모델처럼 동일한 속성을 나타내는 것을 중지 할 가능성이 있다 (?)

그러나 이 경우에서 별로 문제는 X

여기서 제시하는 해결 방법은 미리 학습된 word2vec embedding을 사용하고 embedding layer에 더 낮은 learning rate를 사용하는 것이다. (일반 learning rate에 0.1을 곱함)

**Deep representation**

**Deep representation** 부분의 주 목적은 모든 연관 정보를 output에 응축하는 한편, single sample을 식별할 수 있는 부분을 억제하는 것이다.