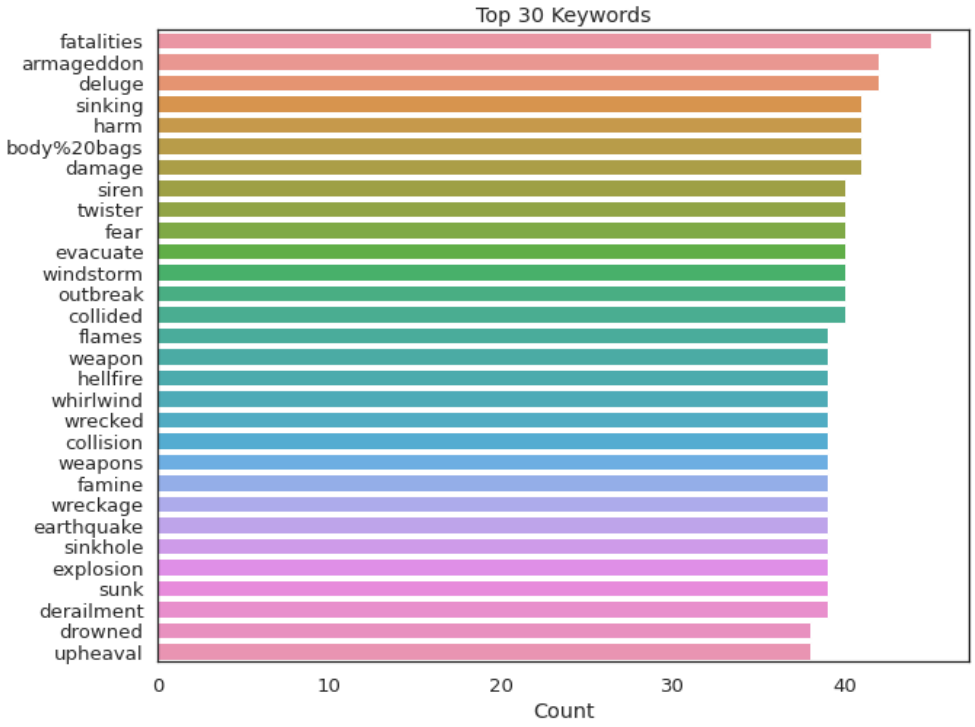
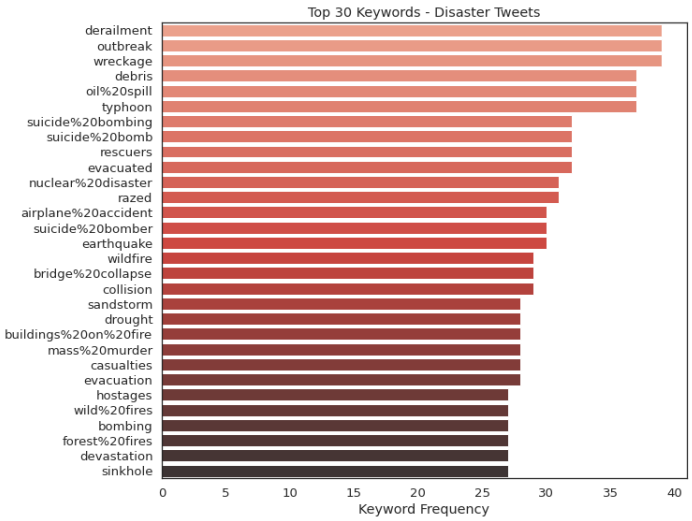
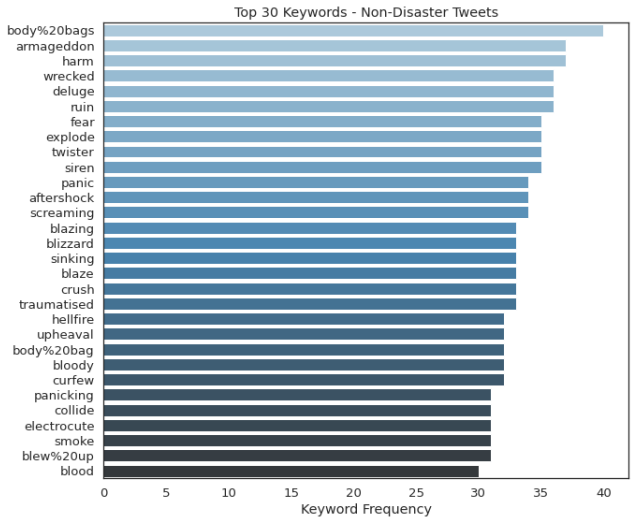
در ابتدا به تحلیل داده‌های سوال می‌پردازیم تا دید بهتری نسبت به مجموعه داده پیدا کنیم.

ستونی تحت عنوان keyword در مجموعه داده وجود دارد که در برخی از توییت‌ها مقدار غیر نال دارد. اگر پرتکرارترین کلمات کلیدی را استخراج کنیم به صورت زیر خواهد بود:

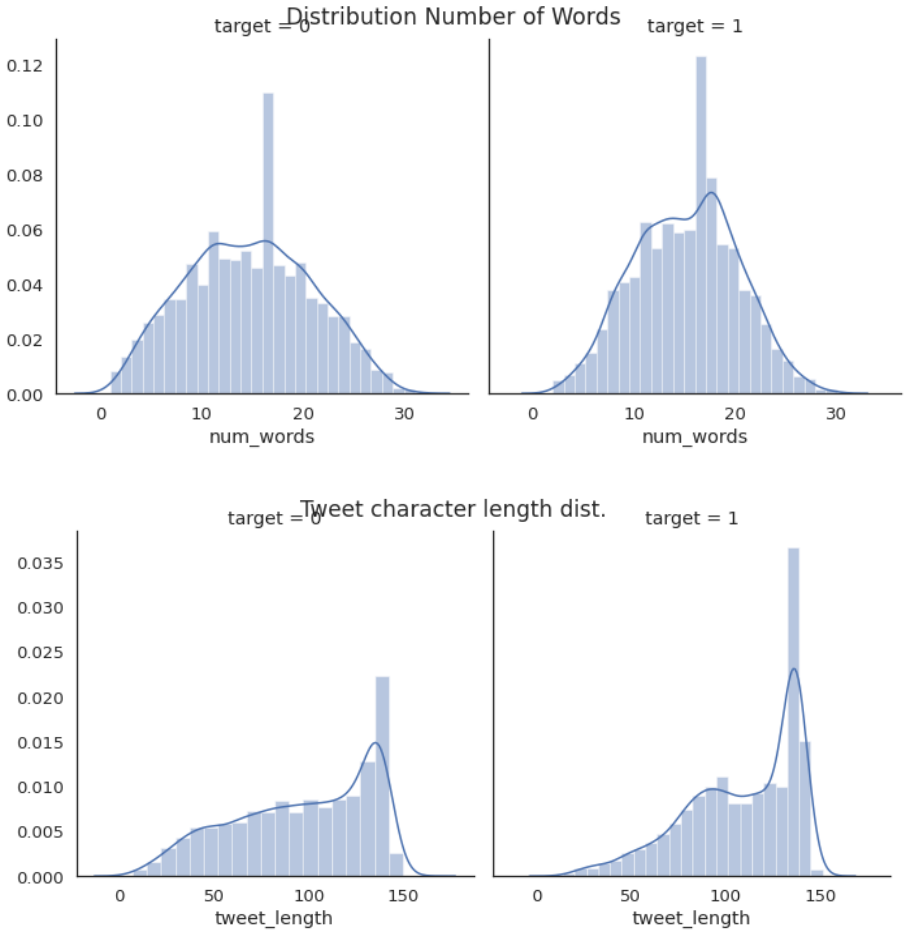


و اگر به تفکیک target کلمات کلیدی پرتکرار را استخراج کنیم به صورت زیر خواهد.

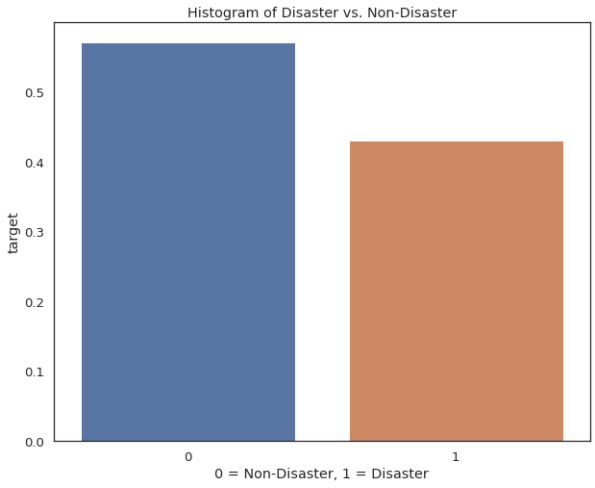




تعداد کاراکترهای توییت‌ها و تعداد کلمات داخل هر توییت نیز برای هر تارگت از توزیع خاصی برخوردار است که در شکل زیر مشاهده میکنیم.



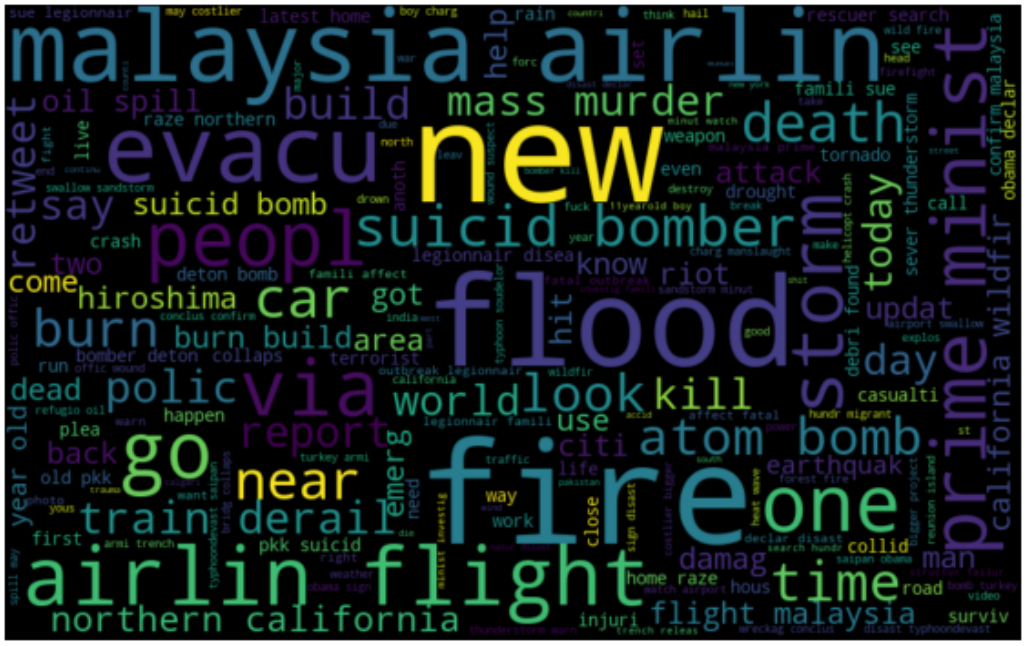
توزیع تعداد توییت‌های از هر تارگت نیز به صورت زیر است.



در مرحله بعد داده‌های متنی را تمیز می‌کنیم تا کلمات یکسان با نوشتارهای متفاوت حتی‌الامکان به یک کلمه تبدیل شوند. با استفاده از رجکس و برخی از توابع حاضر در کتابخانه‌ها، آدرس‌های وبسایت، تگ‌های html، کلمات غیر ASCII، اموجی‌ها، علائم نگارشی از متن حذف می‌شوند. همه کلمات lowercase می‌شوند و با استفاده از کتابخانه contractions و همچنین استفاده از دیکشنری‌های موجود در اینترنت کلمات محاوره‌ای مثل b4 تبدیل به کلمات اصلی (before) می‌شوند.

سپس با استفاده از کتابخانه NLTK کلمات stopword زبان انگلیسی از متن حذف می‌شود تا کلماتی که حاوی اطلاعات خاصی نیستند را از متن حذف کنیم. همچنین با استفاده از همین کتابخانه و استفاده از stemmer و lemmatizer های موجود کلمات حاضر در متن را به ریشه آن کلمات تبدیل می‌کنیم.

پس از حذف کلمات stopwords و نرمال کردن کلمات ابر کلماتی که در توییتر نشان‌دهنده اتفاق بدی بودند به صورت زیر خواهد بود.



در مرحله بعد کلمات را به بردار تبدیل می‌کنیم تا برای مدلی که استفاده می‌کنیم قابل فهم باشد. این کار را با استفاده از کتابخانه gensim و با متد skipgram انجام می‌دهیم. مدل sg را روی تمام متن‌های توییت‌هایی که در اختیار داریم آموزش می‌دهیم. در این مرحله کلمات به یک فضای برداری n بعدی جانمایی می‌شوند که فاصله

اقلیدسی بین آن‌ها معنا دار است. در نتیجه کلمات شبیه به هم در فضا نزدیک هم قرار می‌گیرند. به طور مثال نزدیک‌ترین کلمات به بردار کلمه 'car' کلمات زیر هستند:

```
[('xoxo', 0.8200212121009827),
 ('motorcycl', 0.7866523861885071),
 ('disney', 0.7720800638198853),
 ('rail', 0.7695556282997131),
 ('wreck', 0.7671979665756226),
 ('identitytheft', 0.7542692422866821),
 ('hard', 0.7363719940185547),
 ('aquarium', 0.7363387942314148),
 ('cnbc', 0.735724687576294),
 ('lover', 0.7341086864471436)]
```

در این متد هایپرپارامترهای زیادی داریم که در نهایت روی خروجی مدل تاثیرگذار خواهند بود، از مهم‌ترین این پارامترها می‌توان به موارد زیر اشاره کرد (که با تست کردن مقادیر مختلف در کد مقادیری که نتیجه بهتری می‌دادند انتخاب شدند):

- پارامتر window size: که تعداد کلمات کنار هم مورد بررسی در الگوریتم skip-gram هنگامی که می‌خواهیم کلمه وسط را پیش‌بینی کنیم است.
- پارامتر min count: کلماتی که از کمتر از این مقدار در متن تکرار شدند را به بردار تبدیل نمی‌کنیم.
- سایز فضای برداری: از مهم‌ترین پارامترها سایز بردارهای کلمات است.

در مرحله بعد برای تبدیل کلمات یک توییت به بردار، به جای هر کلمه بردار آن کلمه و اگر کلمه‌ای در فضای برداریمان حاضر نبود، یک بردار خاص (نمونه برداری شده از نرمال استاندارد) به جای آن قرار می‌دهیم. از آنجایی که در مدلی مانند LSTM لازم است طول همه جملات یکسان باشد، در نتیجه با توجه به میانگین و انحراف معیار طول جملات در مجموعه داده یک طول ثابت مانند ۱۲ را فرض می‌کنیم و در توییت‌های طولانی تر از ۱۲ کلمه، کلمات ۱۲ به بعد را لحاظ نمی‌کنیم و برای توییت‌های کوتاه‌تر، از بردار تمام صفر برای رساندن طول توییت به ۱۲ استفاده می‌کنیم.

در نهایت با استفاده از یک LSTM و با بهینه کردن هایپرپارامترهای آن مدلی را برای تشخیص تارگت از روی تکست آموزش می‌دهیم که روی داده تست سایت ککل به دقت ۷۷ درصد رسید. (در مرحله hyperparameter tuning خیلی جای کار بیشتری داشت که به علت کمبود وقت میسر نشد مدل جزو ۳۰ درصد برتر شود).