

REAL OR NOT? NLP WITH DISASTER TWEETS



SHIBA - INU

XUMIN CAI
KIRIL KUZMIN
CHENYU WANG
CHAN AEK PANICHVATANA
PRAGNA REDDY KANCHARLA
KRISHNA MANI TEJA KATRAGADDA

CONTENTS

- The Problem
- Motivation
- Dataset
- Baseline model
- Models
- Commit History
- Results
- Conclusion

THE PROBLEM



- Given: a **text of a tweet** and some metadata (optional)

Metadata:

- Location
- Keyword

- Find: if the tweet talks about real disaster
- TRAIN SIZE: 7613 TEST SIZE: 3263

MOTIVATIONS

- Monitor the situation in a real time
 - Quick
 - Inexpensive
 - Safe
- Predict a follow-up process
- Real situation from contradicting tweets
- Categorization of situational and conversational data is important

PRE-PROCESSING DATA

- Convert text to lowercase
- Remove stopwords
- Remove punctuation
- Remove digits
- Text vectorization (TF-IDF transformation)

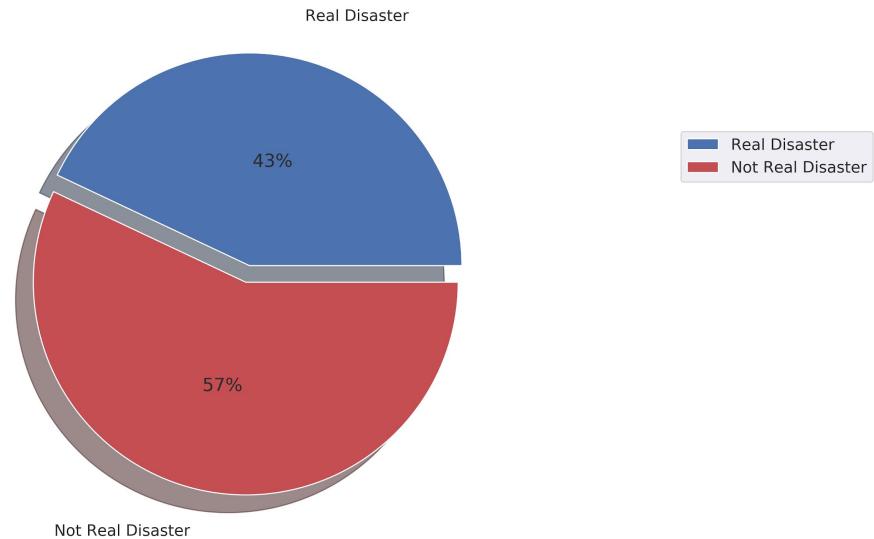
					id	keyword	location	text	target
90	132	accident					NaN	???? it was an accident http://t.co/Oia5fxi4gM	0
91	133	accident	New Hanover County, NC	FYI CAD:FYI: ;ACCIDENT PROPERTY DAMAGE;WPD;160...					1
92	134	accident					NaN	8/6/2015@2:09 PM: TRAFFIC ACCIDENT NO INJURY a...	1



affect	aftershock	air	airplan	airplaneaccid	airport	...	wreck	wreckag	year	yet	your	youtub
0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.00000	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.29624	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.00000	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.00000	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.00000	0.0	0.0	0.0
...
0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.00000	0.0	0.0	0.0

DATA RE-EXPLORATION

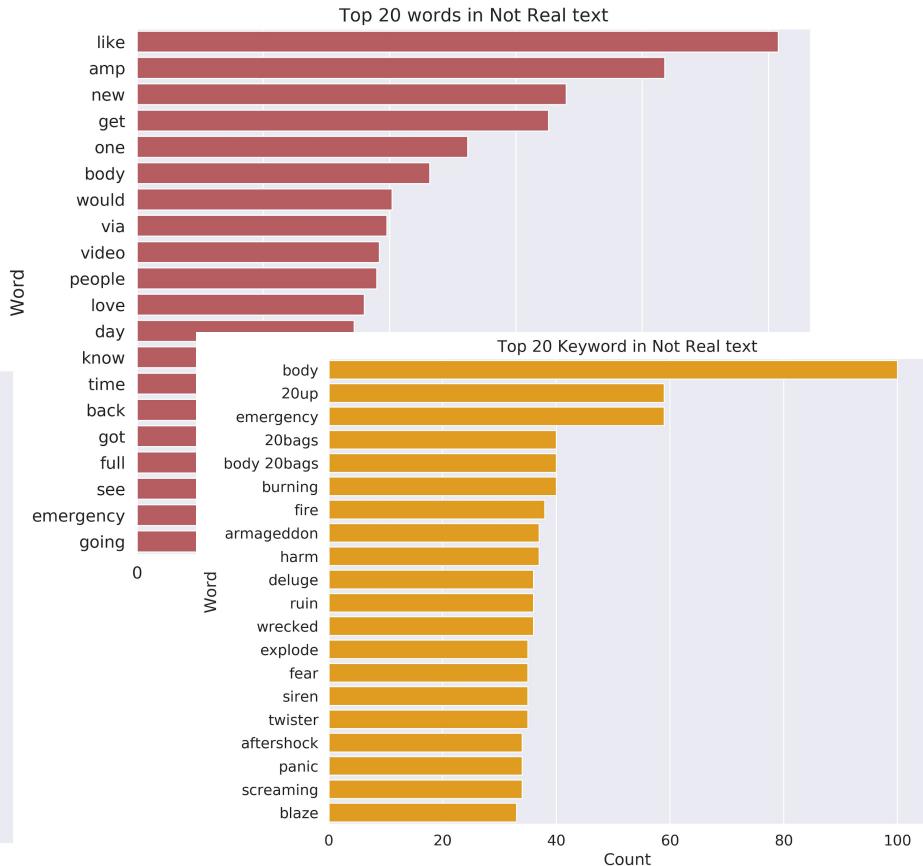
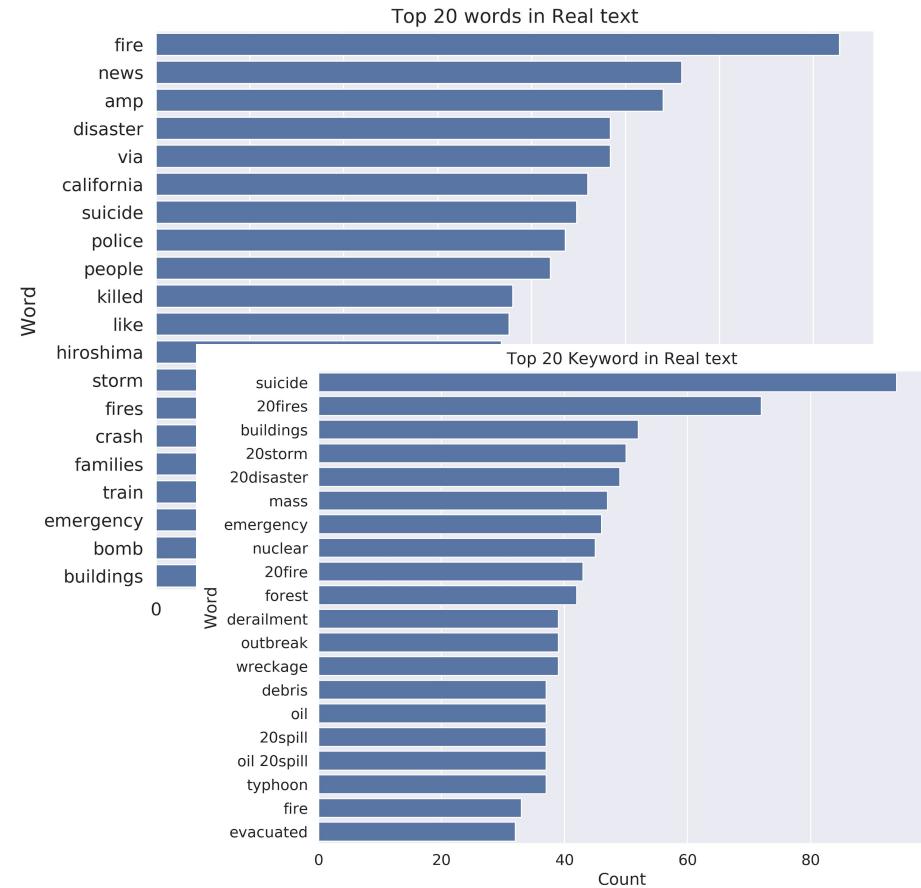
- Frequent words analysis
 - Hyperlink Analysis
 - Vocabulary Embedding
- Coverage Analysis



Label 1: 3271 of 7613 twits

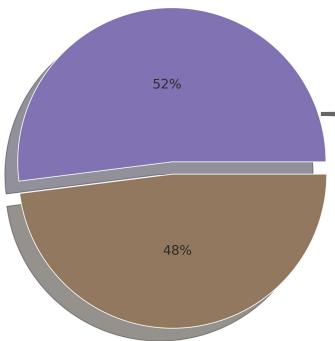
Label 0: 4342 of 7613 twits

TOP 20 FREQUENT WORDS



HYPERLINK ANALYSIS

Twitter contains hyperlink

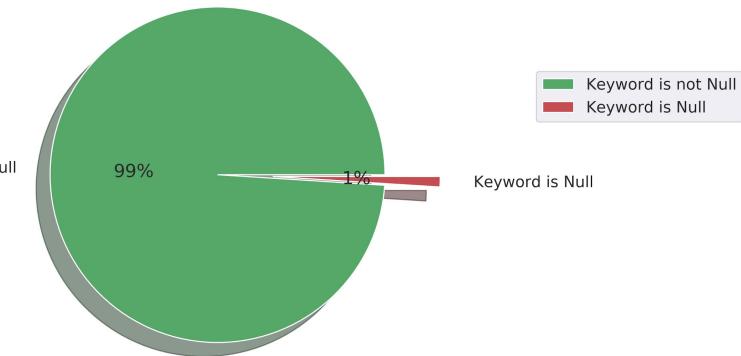
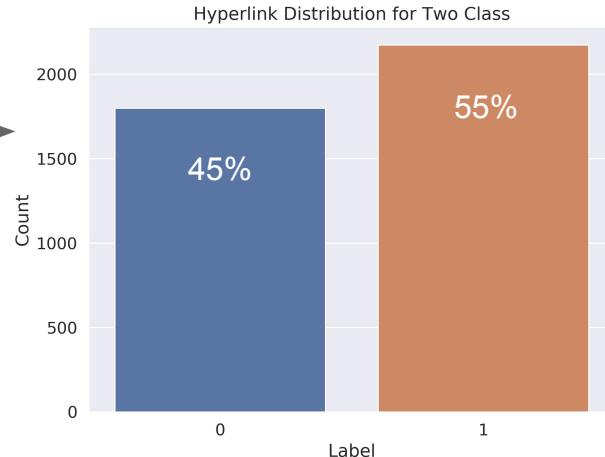


Twitter Not contains hyperlink

3971 out of 7613 training data
contains hyperlink
“<http://t.co/lHYXE0HY6C>”

2172 out of 3971 is label 1
1799 out of 3971 is label 0

3951 out of 3971 contain keyword
20 out of 3971 don't have keyword



EQUIVALENT CLASSES

- ★ 336 equivalent classes (1092 tweets out of 7613)
- ★ 79 “bad” equivalent classes (18 from the beginning)

{40, 48} {104, 106, 114, 115, 116, 118, 119} {113, 131} {139, 159} {154, 143} {147, 164} {197, 201, 202, 172, 207, 177, 183, 190, 191} {186, 193, 178, 192} {203, 204} {216, 228} {218, 220, 238} {259, 266, 270, 271, 246, 250, 251, 253} {248, 269} {364, 370, 371, 372, 373, 346, 349, 350} {358, 360, 365, 378, 347, 351} {379, 367} {376, 369, 377} {386, 387, 403, 405} {393, 394} {405, 407} {410, 411} {402, 392, 406, 404} {419, 420} {471, 472} {480, 481} {514, 514} {516, 519} {509, 49, 492, 49, 499, 10, 504, 505} {506, 508, 510, 511} {569, 547} {593, 550, 2604, 2607, 2611, 2582, 270, 2590, 2591} {597, 574} {570, 60, 584, 608} {597, 591} {592, 603, 604, 606} {66, 59} {624, 630, 630, 634} {682, 683} {795, 788} {952, 953, 956} {981, 972, 957, 958} {970, 963} {966, 983} {969, 981, 974} {1045, 1022} {1064, 1096} {1075, 1070} {1133, 1134} {1144, 1145, 1146, 1147, 1148} {1160, 1193} {1186, 1187} {1204, 1205, 1209, 1210, 1211, 1212} {1223} {1317, 1318} {1399, 1400} {1402, 1403, 1404} {1411, 1412} {1422} {1232, 1230} {1242, 1251} {1284, 1277} {1386, 1370, 1390, 1374} {1384, 137} {1425, 1420} {1506, 1510, 1516, 1518, 1501, 1502, 1503} {1505, 1515} {1514, 1522, 1519} {1528, 153} {1548, 1542} {1549, 1571} {1592} {155, 1500} {1618, 1614} {16, 166, 166} {163} {1761} {1765} {1704, 1725} {1732, 1719} {1739, 1749, 1750} {1761, 1773} {4601, 1771} {1882, 1883, 1892, 188} {1890, 1891, 1892} {1893, 1894} {1895, 1896} {1897, 1898} {1899, 189} {1900, 1901} {1902, 1903} {1904, 1905} {1906, 1907} {1907, 1908} {1909, 1910} {1910, 1911} {1911, 1912} {1912, 1913} {1913, 1914} {1914, 1915} {1915, 1916} {1916, 1917} {1917, 1918} {1918, 1919} {1919, 1920} {1920, 1921} {1921, 1922} {1922, 1923} {1923, 1924} {1924, 1925} {1925, 1926} {1926, 1927} {1927, 1928} {1928, 1929} {1929, 1930} {1930, 1931} {1931, 1932} {1932, 1933} {1933, 1934} {1934, 1935} {1935, 1936} {1936, 1937} {1937, 1938} {1938, 1939} {1939, 1940} {1940, 1941} {1941, 1942} {1942, 1943} {1943, 1944} {1944, 1945} {1945, 1946} {1946, 1947} {1947, 1948} {1948, 1949} {1949, 1950} {1950, 1951} {1951, 1952} {1952, 1953} {1953, 1954} {1954, 1955} {1955, 1956} {1956, 1957} {1957, 1958} {1958, 1959} {1959, 1960} {1960, 1961} {1961, 1962} {1962, 1963} {1963, 1964} {1964, 1965} {1965, 1966} {1966, 1967} {1967, 1968} {1968, 1969} {1969, 1970} {1970, 1971} {1971, 1972} {1972, 1973} {1973, 1974} {1974, 1975} {1975, 1976} {1976, 1977} {1977, 1978} {1978, 1979} {1979, 1980} {1980, 1981} {1981, 1982} {1982, 1983} {1983, 1984} {1984, 1985} {1985, 1986} {1986, 1987} {1987, 1988} {1988, 1989} {1989, 1990} {1990, 1991} {1991, 1992} {1992, 1993} {1993, 1994} {1994, 1995} {1995, 1996} {1996, 1997} {1997, 1998} {1998, 1999} {1999, 190} {190, 323} {323, 324} {324, 325} {325, 326} {326, 327} {327, 328} {328, 329} {329, 330} {330, 331} {331, 332} {332, 333} {333, 334} {334, 335} {335, 336} {336, 337} {337, 338} {338, 339} {339, 340} {340, 341} {341, 342} {342, 343} {343, 344} {344, 345} {345, 346} {346, 347} {347, 348} {348, 349} {349, 350} {350, 351} {351, 352} {352, 353} {353, 354} {354, 355} {355, 356} {356, 357} {357, 358} {358, 359} {359, 360} {360, 361} {361, 362} {362, 363} {363, 364} {364, 365} {365, 366} {366, 367} {367, 368} {368, 369} {369, 370} {370, 371} {371, 372} {372, 373} {373, 374} {374, 375} {375, 376} {376, 377} {377, 378} {378, 379} {379, 380} {380, 381} {381, 382} {382, 383} {383, 384} {384, 385} {385, 386} {386, 387} {387, 388} {388, 389} {389, 390} {390, 391} {391, 392} {392, 393} {393, 394} {394, 395} {395, 396} {396, 397} {397, 398} {398, 399} {399, 30} {30, 31} {31, 32} {32, 33} {33, 34} {34, 35} {35, 36} {36, 37} {37, 38} {38, 39} {39, 40} {40, 41} {41, 42} {42, 43} {43, 44} {44, 45} {45, 46} {46, 47} {47, 48} {48, 49} {49, 50} {50, 51} {51, 52} {52, 53} {53, 54} {54, 55} {55, 56} {56, 57} {57, 58} {58, 59} {59, 60} {60, 61} {61, 62} {62, 63} {63, 64} {64, 65} {65, 66} {66, 67} {67, 68} {68, 69} {69, 70} {70, 71} {71, 72} {72, 73} {73, 74} {74, 75} {75, 76} {76, 77} {77, 78} {78, 79} {79, 80} {80, 81} {81, 82} {82, 83} {83, 84} {84, 85} {85, 86} {86, 87} {87, 88} {88, 89} {89, 90} {90, 91} {91, 92} {92, 93} {93, 94} {94, 95} {95, 96} {96, 97} {97, 98} {98, 99} {99, 100} {100, 101} {101, 102} {102, 103} {103, 104} {104, 105} {105, 106} {106, 107} {107, 108} {108, 109} {109, 110} {110, 111} {111, 112} {112, 113} {113, 114} {114, 115} {115, 116} {116, 117} {117, 118} {118, 119} {119, 120} {120, 121} {121, 122} {122, 123} {123, 124} {124, 125} {125, 126} {126, 127} {127, 128} {128, 129} {129, 130} {130, 131} {131, 132} {132, 133} {133, 134} {134, 135} {135, 136} {136, 137} {137, 138} {138, 139} {139, 140} {140, 141} {141, 142} {142, 143} {143, 144} {144, 145} {145, 146} {146, 147} {147, 148} {148, 149} {149, 150} {150, 151} {151, 152} {152, 153} {153, 154} {154, 155} {155, 156} {156, 157} {157, 158} {158, 159} {159, 160} {160, 161} {161, 162} {162, 163} {163, 164} {164, 165} {165, 166} {166, 167} {167, 168} {168, 169} {169, 170} {170, 171} {171, 172} {172, 173} {173, 174} {174, 175} {175, 176} {176, 177} {177, 178} {178, 179} {179, 180} {180, 181} {181, 182} {182, 183} {183, 184} {184, 185} {185, 186} {186, 187} {187, 188} {188, 189} {189, 190} {190, 191} {191, 192} {192, 193} {193, 194} {194, 195} {195, 196} {196, 197} {197, 198} {198, 199} {199, 200} {200, 201} {201, 202} {202, 203} {203, 204} {204, 205} {205, 206} {206, 207} {207, 208} {208, 209} {209, 210} {210, 211} {211, 212} {212, 213} {213, 214} {214, 215} {215, 216} {216, 217} {217, 218} {218, 219} {219, 220} {220, 221} {221, 222} {222, 223} {223, 224} {224, 225} {225, 226} {226, 227} {227, 228} {228, 229} {229, 230} {230, 231} {231, 232} {232, 233} {233, 234} {234, 235} {235, 236} {236, 237} {237, 238} {238, 239} {239, 240} {240, 241} {241, 242} {242, 243} {243, 244} {244, 245} {245, 246} {246, 247} {247, 248} {248, 249} {249, 250} {250, 251} {251, 252} {252, 253} {253, 254} {254, 255} {255, 256} {256, 257} {257, 258} {258, 259} {259, 260} {260, 261} {261, 262} {262, 263} {263, 264} {264, 265} {265, 266} {266, 267} {267, 268} {268, 269} {269, 270} {270, 271} {271, 272} {272, 273} {273, 274} {274, 275} {275, 276} {276, 277} {277, 278} {278, 279} {279, 280} {280, 281} {281, 282} {282, 283} {283, 284} {284, 285} {285, 286} {286, 287} {287, 288} {288, 289} {289, 290} {290, 291} {291, 292} {292, 293} {293, 294} {294, 295} {295, 296} {296, 297} {297, 298} {298, 299} {299, 300} {300, 301} {301, 302} {302, 303} {303, 304} {304, 305} {305, 306} {306, 307} {307, 308} {308, 309} {309, 310} {310, 311} {311, 312} {312, 313} {313, 314} {314, 315} {315, 316} {316, 317} {317, 318} {318, 319} {319, 320} {320, 321} {321, 322} {322, 323} {323, 324} {324, 325} {325, 326} {326, 327} {327, 328} {328, 329} {329, 330} {330, 331} {331, 332} {332, 333} {333, 334} {334, 335} {335, 336} {336, 337} {337, 338} {338, 339} {339, 340} {340, 341} {341, 342} {342, 343} {343, 344} {344, 345} {345, 346} {346, 347} {347, 348} {348, 349} {349, 350} {350, 351} {351, 352} {352, 353} {353, 354} {354, 355} {355, 356} {356, 357} {357, 358} {358, 359} {359, 360} {360, 361} {361, 362} {362, 363} {363, 364} {364, 365} {365, 366} {366, 367} {367, 368} {368, 369} {369, 370} {370, 371} {371, 372} {372, 373} {373, 374} {374, 375} {375, 376} {376, 377} {377, 378} {378, 379} {379, 380} {380, 381} {381, 382} {382, 383} {383, 384} {384, 385} {385, 386} {386, 387} {387, 388} {388, 389} {389, 390} {390, 391} {391, 392} {392, 393} {393, 394} {394, 395} {395, 396} {396, 397} {397, 398} {398, 399} {399, 400} {400, 401} {401, 402} {402, 403} {403, 404} {404, 405} {405, 406} {406, 407} {407, 408} {408, 409} {409, 410} {410, 411} {411, 412} {412, 413} {413, 414} {414, 415} {415, 416} {416, 417} {417, 418} {418, 419} {419, 420} {420, 421} {421, 422} {422, 423} {423, 424} {424, 425} {425, 426} {426, 427} {427, 428} {428, 429} {429, 430} {430, 431} {431, 432} {432, 433} {433, 434} {434, 435} {435, 436} {436, 437} {437, 438} {438, 439} {439, 440} {440, 441} {441, 442} {442, 443} {443, 444} {444, 445} {445, 446} {446, 447} {447, 448} {448, 449} {449, 450} {450, 451} {451, 452} {452, 453} {453, 454} {454, 455} {455, 456} {456, 457} {457, 458} {458, 459} {459, 460} {460, 461} {461, 462} {462, 463} {463, 464} {464, 465} {465, 466} {466, 467} {467, 468} {468, 469} {469, 470} {470, 471} {471, 472} {472, 473} {473, 474} {474, 475} {475, 476} {476, 477} {477, 478} {478, 479} {479, 480} {480, 481} {481, 482} {482, 483} {483, 484} {484, 485} {485, 486} {486, 487} {487, 488} {488, 489} {489, 490} {490, 491} {491, 492} {492, 493} {493, 494} {494, 495} {495, 496} {496, 497} {497, 498} {498, 499} {499, 500} {500, 501} {501, 502} {502, 503} {503, 504} {504, 505} {505, 506} {506, 507} {507, 508} {508, 509} {509, 510} {510, 511} {511, 512} {512, 513} {513, 514} {514, 515} {515, 516} {516, 517} {517, 518} {518, 519} {519, 520} {520, 521} {521, 522} {522, 523} {523, 524} {524, 525} {525, 526} {526, 527} {527, 528} {528, 529} {529, 530} {530, 531} {531, 532} {532, 533} {533, 534} {534, 535} {535, 536} {536, 537} {537, 538} {538, 539} {539, 540} {540, 541} {541, 542} {542, 543} {543, 544} {544, 545} {545, 546} {546, 547} {547, 548} {548, 549} {549, 550} {550, 551} {551, 552} {552, 553} {553, 554} {554, 555} {555, 556} {556, 557} {557, 558} {558, 559} {559, 560} {560, 561} {561, 562} {562, 563} {563, 564} {564, 565} {565, 566} {566, 567} {567, 568} {568, 569} {569, 570} {570, 571} {571, 572} {572, 573} {573, 574} {574, 575} {575, 576} {576, 577} {577, 578} {578, 579} {579, 580} {580, 581} {581, 582} {582, 583} {583, 584} {584, 585} {585, 586} {586, 587} {587, 588} {588, 589} {589, 590} {590, 591} {591, 592} {592, 593} {593, 594} {594, 595} {595, 596} {596, 597} {597, 598} {598, 599} {599, 600} {600, 601} {601, 602} {602, 603} {603, 604} {604, 605} {605, 606} {606, 607} {607, 608} {608, 609} {609, 610} {610, 611} {611, 612} {612, 613} {613, 614} {614, 615} {615, 616} {616, 617} {617, 618} {618, 619} {619, 620} {620, 621} {621, 622} {622, 623} {623, 624} {624, 625} {625, 626} {626, 627} {627, 628} {628, 629} {629, 630} {630, 631} {631, 632} {632, 633} {633, 634} {634, 635} {635, 636} {636, 637} {637, 638} {638, 639} {639, 640} {640, 641} {641, 642} {642, 643} {643, 644} {644, 645} {645, 646} {646, 647} {647, 648} {648, 649} {649, 650} {650, 651} {651, 652} {652, 653} {653, 654} {654, 655} {655, 656} {656, 657} {657, 658} {658, 659} {659, 660} {660, 661} {661, 662} {662, 663} {663, 664} {664, 665} {665, 666} {666, 667} {667, 668} {668, 669} {669, 670} {670, 671} {671, 672} {672, 673} {673, 674} {674, 675} {675, 676} {676, 677} {677, 678} {678, 679} {679, 680} {680, 681} {681, 682} {682, 683} {683, 684} {684, 685} {685, 686} {686, 687} {687, 688} {688, 689} {689, 690} {690, 691} {691, 692} {692, 693} {693, 694} {694, 695} {695, 696} {696, 697} {697, 698} {698, 699} {699, 700} {700, 701} {701, 702} {702, 703} {703, 704} {704, 705} {705, 706} {706, 707} {707, 708} {708, 709} {709, 710} {710, 711} {711, 712} {712, 713} {713, 714} {714, 715} {715, 716} {716, 717} {717, 718} {718, 719} {719, 720} {720, 721} {721, 722} {722, 723} {723, 724} {724, 725} {725, 726} {726, 727} {727, 728} {728, 729} {729, 730} {730, 731} {731, 732} {732, 733} {733, 734} {734, 735} {735, 736} {736, 737} {737, 738} {738, 739} {739, 740} {740, 741} {741, 742} {742, 743} {743, 744} {744, 745} {745, 746} {746, 747} {747, 748} {748, 749} {749, 750} {750, 751} {751, 752} {752, 753} {753, 754} {754, 755} {755, 756} {756, 757} {757, 758} {758, 759} {759, 760} {760, 761} {761, 762} {762, 763} {763, 764} {764, 765} {765, 766} {766, 767} {767, 768} {768, 769} {769, 770} {770, 771} {771, 772} {772, 773} {773, 774} {774, 775} {775, 776} {776, 777} {777, 778} {778, 779} {779, 780} {780, 781} {781, 782} {782, 783} {783, 784} {784, 785} {785, 786} {786, 787} {787, 788} {788, 789} {789, 790} {790, 791} {791, 792} {792, 793} {793, 794} {794, 795} {795, 796} {796, 797} {797, 798} {798, 799} {799, 800} {800, 801} {801, 802} {802, 803} {803, 804} {804, 805} {805, 806} {806, 807} {807, 808} {808, 809} {809, 810} {810, 811} {811, 812} {812, 813} {813, 814} {814, 815} {815, 816} {816, 817} {817, 818} {818, 819} {819, 820} {820, 821} {821, 822} {822, 823} {823, 824} {824, 825} {825, 826} {826, 827} {827, 828} {828, 829} {829, 830} {830, 831} {831, 832} {832, 833} {833, 834} {834, 835} {835, 836} {836, 837} {837, 838} {838, 839} {839, 840} {840, 841} {841, 842} {842, 843} {843, 844} {844, 845} {845, 846} {846, 847} {847, 848} {848, 849} {849, 850} {850, 851} {851, 852} {852, 853} {853, 854} {854, 855} {855, 856} {856, 857} {857, 858} {858, 859} {859, 860} {860, 861} {861, 862} {862, 863} {863, 864} {864, 865} {865, 866} {866, 867} {867, 868} {868, 869} {869, 870} {870, 871} {871, 872} {872, 873} {873, 874} {874, 875} {875, 876} {876, 877} {877, 878} {878, 879} {879, 880} {880, 881} {881, 882} {882, 883} {883, 884} {884, 885} {885, 886} {886, 887} {887, 888} {888, 889} {889, 890} {890, 891} {891, 892} {892, 893} {893, 894} {894, 895} {895, 896} {896, 897} {897, 898} {898, 899} {899, 900} {900, 901} {901, 902} {902, 903} {903, 904} {904, 905} {905, 906} {906, 907} {907, 908} {908, 909} {909, 910} {910, 911} {911, 912} {912, 913} {913, 914} {914, 915} {915, 916} {916, 917} {917, 918} {918, 919} {919, 920} {920, 921} {921, 922} {922, 923} {923, 924} {924, 925} {925, 926} {926, 927} {927, 928} {928, 929} {929, 930} {930, 931} {931, 932} {932, 933} {933, 934} {934, 935} {935, 936} {936, 937} {937, 938} {938, 939} {939, 9

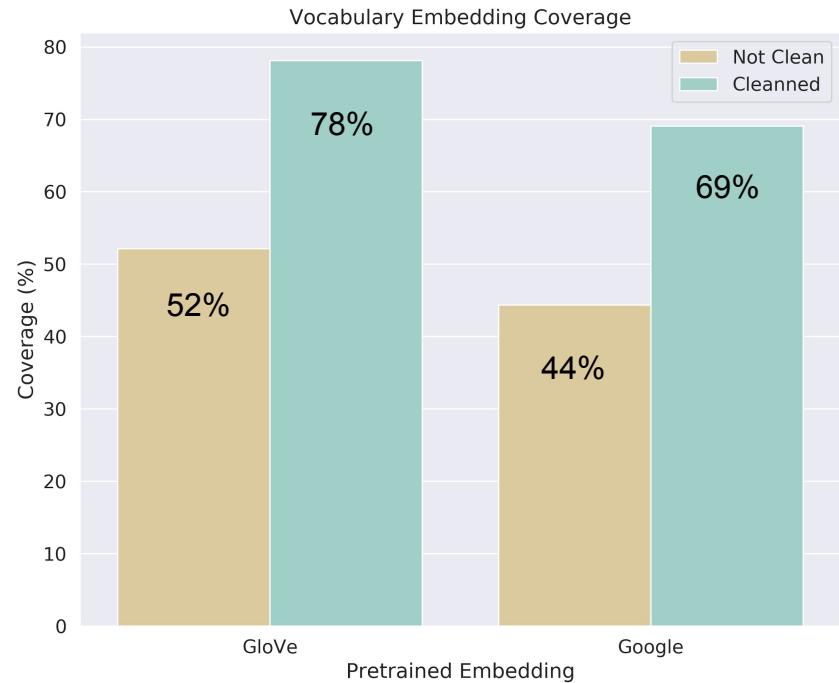
VOCABULARY EMBEDDING COVERAGE

- GloVe (300d)
- Google Negative(300d)

Some of the words not included in the embedding dictionary.

Eg. mh370 (GloVe,Google)
→ aircraft accident.

hiroshima, fukushima(Google)
→ atomic bomb



BASELINE MODEL

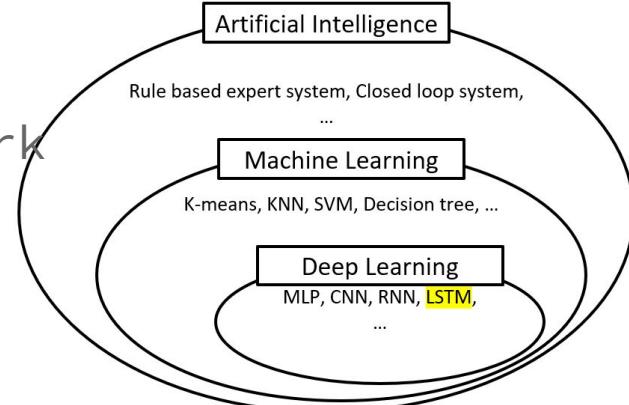
- Vectorize the Text and Calculate the Frequency of each word
 - `from sklearn.feature_extraction.text import TfidfVectorizer`
- Collect Features from `TfidfVectorizer` (without semantics)
(Term Frequency-Inverse Document Frequency)
$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$
- Model : Logistic Regression and Support Vector Machine

	Logistic Regression (no semantic)	SVM (no semantic)
Accuracy	0.7615	0.7815
F1	0.7561	0.7775
Precision	0.7745	0.7845
Recall	0.7611	0.7817

MODELS

LONG SHORT TERM MEMORY - LSTM

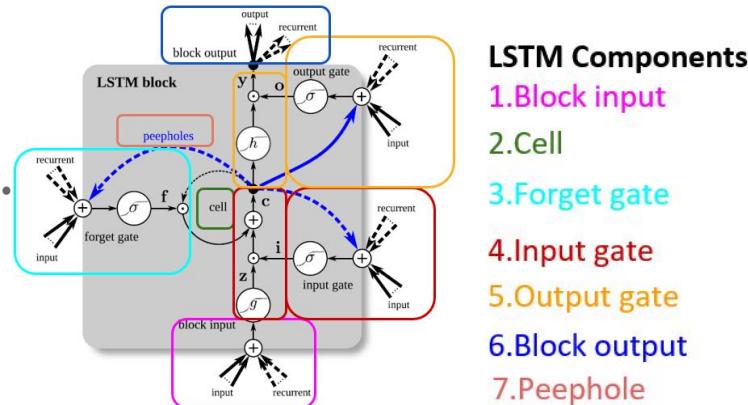
- LSTM is part of Recurrent Neural Network under Deep Learning.



- LSTM architecture has gates and cell.

- LSTM is one of top Kaggle's prizes.
 - LSTM 23%
 - CNN 21%
 - Deep Learning 8%

Reference: Graves and Schmidhuber (2005)



LSTM - MODEL DESIGN



Embedding

LSTM

Dropout

Dense

Activation

```
#LSTM
model_2 = Sequential()
model_2.add(Embedding(max_features,
                     embed_size,
                     weights=[embedding_matrix_glove],
                     input_length=maxlen,
                     trainable=False))
model_2.add(LSTM(32, kernel_regularizer=l2(0.01),
                 recurrent_regularizer=l2(0.01),
                 bias_regularizer=l2(0.01)))
model_2.add(Dropout(0.5))
model_2.add(Dense(units=300, activation='sigmoid'))
model_2.add(Dense(units=100, activation='sigmoid'))
model_2.add(Dense(units=1, activation='sigmoid'))
model_2.compile(loss='binary_crossentropy',
                 optimizer='adam',
                 metrics=['accuracy'])
```

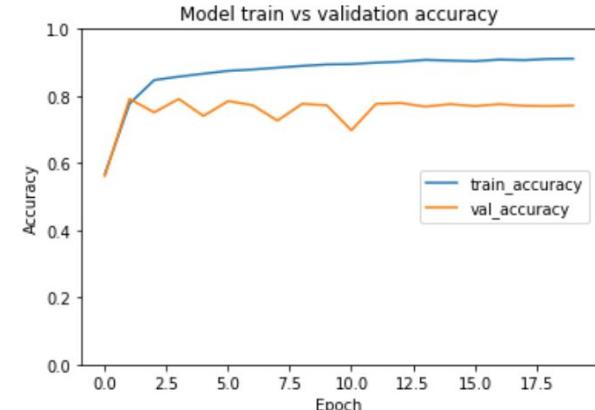
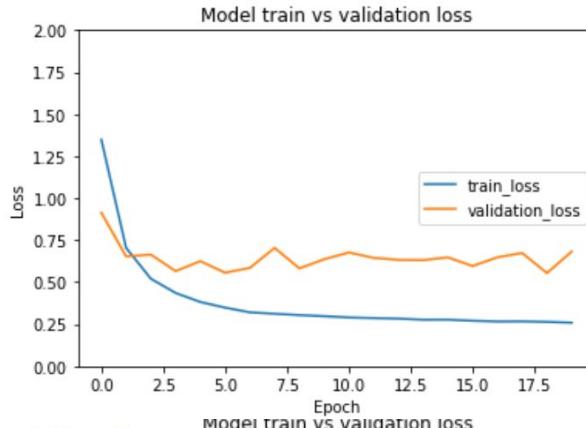


```
Model: "sequential_1"
-----
Layer (type)          Output Shape       Param #
-----
embedding_1 (Embedding)    (None, 26, 300)     3987600
-----
lstm_1 (LSTM)           (None, 32)          42624
-----
dropout_1 (Dropout)      (None, 32)          0
-----
dense_1 (Dense)          (None, 300)         9900
-----
dense_2 (Dense)          (None, 100)        30100
-----
dense_3 (Dense)          (None, 1)           101
-----
Total params: 4,070,325
Trainable params: 82,725
Non-trainable params: 3,987,600
-----
None
```

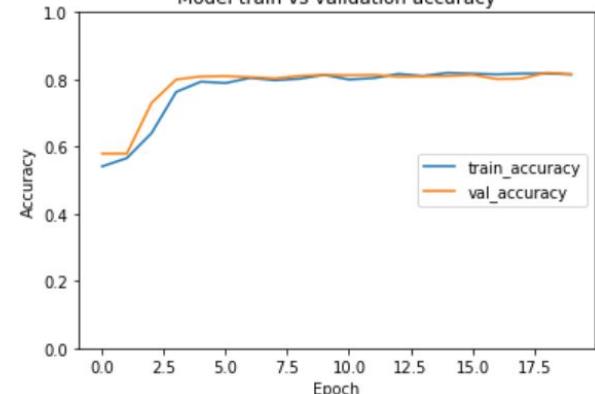
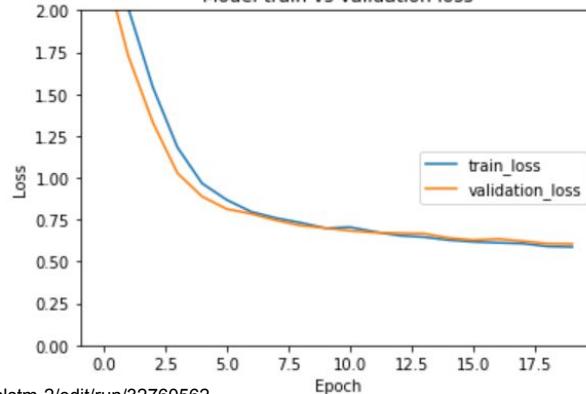
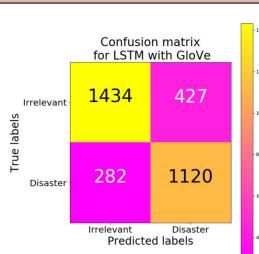
LSTM-MODEL PERFORMANCE

Shallow LSTM : 77%

(Note: Bi-LSTM and GRU show similar result.)



GloVe LSTM : 78%



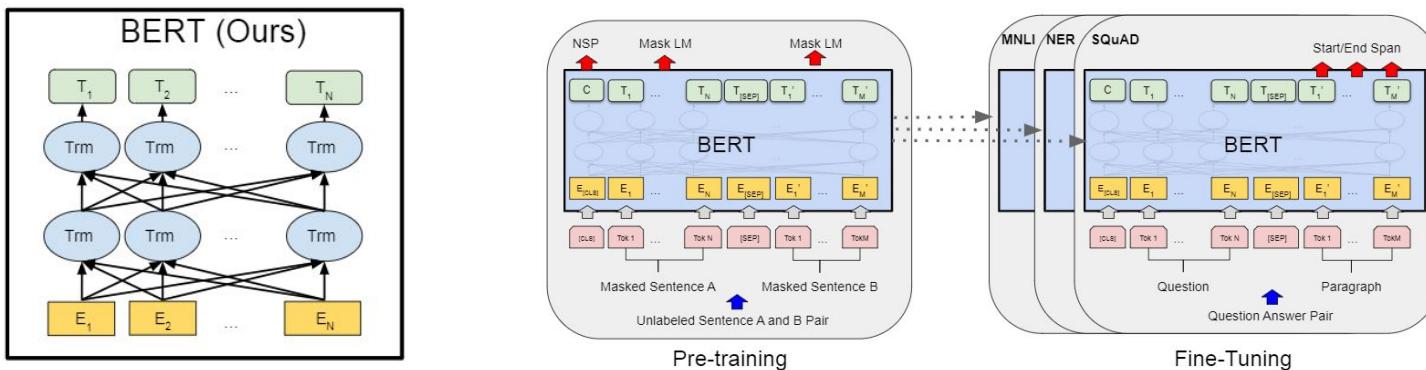
References:

<https://www.kaggle.com/martofthenorth/fork-lstm-of-shiba-inu-keraslstm-2/edit/run/32760562>

<https://www.kaggle.com/martofthenorth/clean1-glove1-lstm1-112-fm-e20-mart?scriptVersionId=32828964>

BERT - BASICS

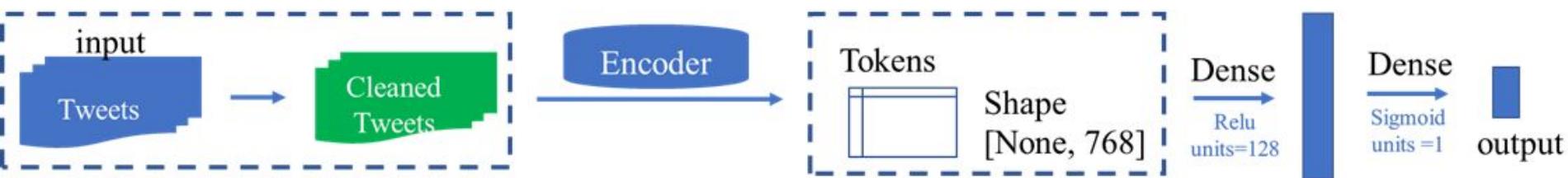
- BERT (Bidirectional Encoder Representations from Transformers)
 - “Masked language model” task
 - “Next sentence Prediction task”



- Encoder: Google pre-trained Model (`bert_en_uncased_L-12_H-768_A-12`)
 - $L = 12$ hidden layers (i.e., Transformer blocks),
 - hidden size : $H = 768$, and $A = 12$ attention heads.

BERT - APPLICATIONS

- Clean Methods
 - clean while keep the original semantics
 - clean while keep the sentence structure
- How we use BERT
 - Original Input: Tweet (Cleaned Version)
 - Input: Result from pre-train BERT is a $[*, 768]$ vector
 - Add 2 Dense Layers
 - Unit = 128, activation Relu/Sigmoid
 - Unit = 1, activation Sigmoid (Final Result)



BERT - TUNING

- Parameters Tuning:
 - Learning Rate: 6e-6
 - Number of Dense Layer : 2
 - Epoch: 3
 - Batch Size: 16
- Result on Testing (w/o clean)
 - with clean:
loss: 0.3158 acc: 0.8774 f1: 0.8474
precision_m: 0.8759 recall: 0.8276
 - Without clean:
loss: 0.4020 acc: **0.8314** f1: 0.7920
precision_m: 0.8689 recall: 0.7387

```
Train on 4263 samples, validate on 1066 samples
Epoch 1/10
4263/4263 [=====] - 114s 27ms/sample - loss: 0.4914 - accuracy: 0.7694
4167 - val_accuracy: 0.8311 - val_f1_m: 0.7739 - val_precision_m: 0.8932 - val_recall_m: 0.7053
Epoch 2/10
4263/4263 [=====] - 97s 23ms/sample - loss: 0.3526 - accuracy: 0.8550
108 - val_accuracy: 0.8330 - val_f1_m: 0.7898 - val_precision_m: 0.8529 - val_recall_m: 0.7616
Epoch 3/10
4263/4263 [=====] - 97s 23ms/sample - loss: 0.2923 - accuracy: 0.8846
543 - val_accuracy: 0.8349 - val_f1_m: 0.7783 - val_precision_m: 0.8923 - val_recall_m: 0.7128
Epoch 4/10
4263/4263 [=====] - 97s 23ms/sample - loss: 0.2367 - accuracy: 0.9062
627 - val_accuracy: 0.8218 - val_f1_m: 0.7641 - val_precision_m: 0.8589 - val_recall_m: 0.7152
Epoch 5/10
4263/4263 [=====] - 97s 23ms/sample - loss: 0.1858 - accuracy: 0.9313
357 - val_accuracy: 0.8255 - val_f1_m: 0.7692 - val_precision_m: 0.8730 - val_recall_m: 0.7087
Epoch 6/10
4263/4263 [=====] - 97s 23ms/sample - loss: 0.1439 - accuracy: 0.9475
933 - val_accuracy: 0.8218 - val_f1_m: 0.7597 - val_precision_m: 0.8738 - val_recall_m: 0.6926
Epoch 7/10
4263/4263 [=====] - 97s 23ms/sample - loss: 0.1085 - accuracy: 0.9615
015 - val_accuracy: 0.8189 - val_f1_m: 0.7703 - val_precision_m: 0.8315 - val_recall_m: 0.7417
Epoch 8/10
4263/4263 [=====] - 97s 23ms/sample - loss: 0.0933 - accuracy: 0.9636
695 - val_accuracy: 0.8086 - val_f1_m: 0.7462 - val_precision_m: 0.8399 - val_recall_m: 0.6901
```

After the deduction of the costs of investing, beating the stock market is a loser's game.

WORD2VEC



CBOW

Skip-gram

Window size of 5



Word Embedding

Word Embedding
Representation
of Input

Dense with
Softmax

Output



(# of unique words, Word embedding size)

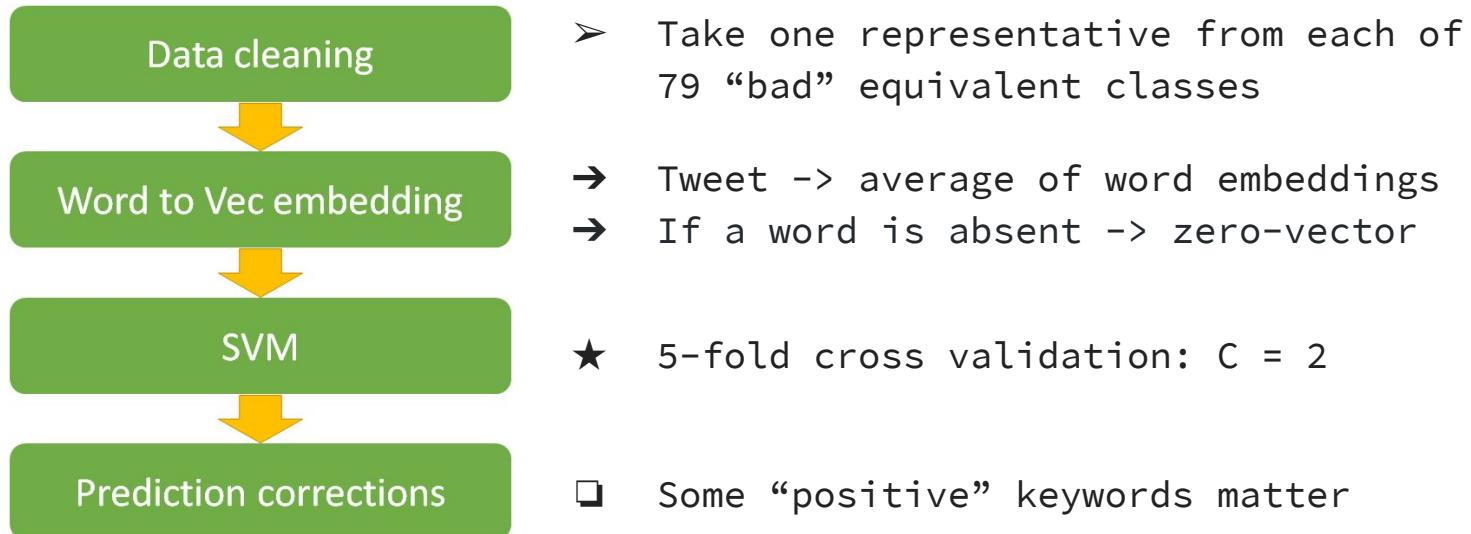
(# of unique words, Word embedding size)

(Word embedding size, 1)

(# of unique words, 1)

GOOGLENEWS-VECTORS-NEGATIVE300

- Pre-trained **vectors** trained on a part of **Google News** dataset (~ 100 billion words)
- Contains 3 million English words mapped to 300-dimension vectors



WORD2VEC: RESULTS

TRAINING

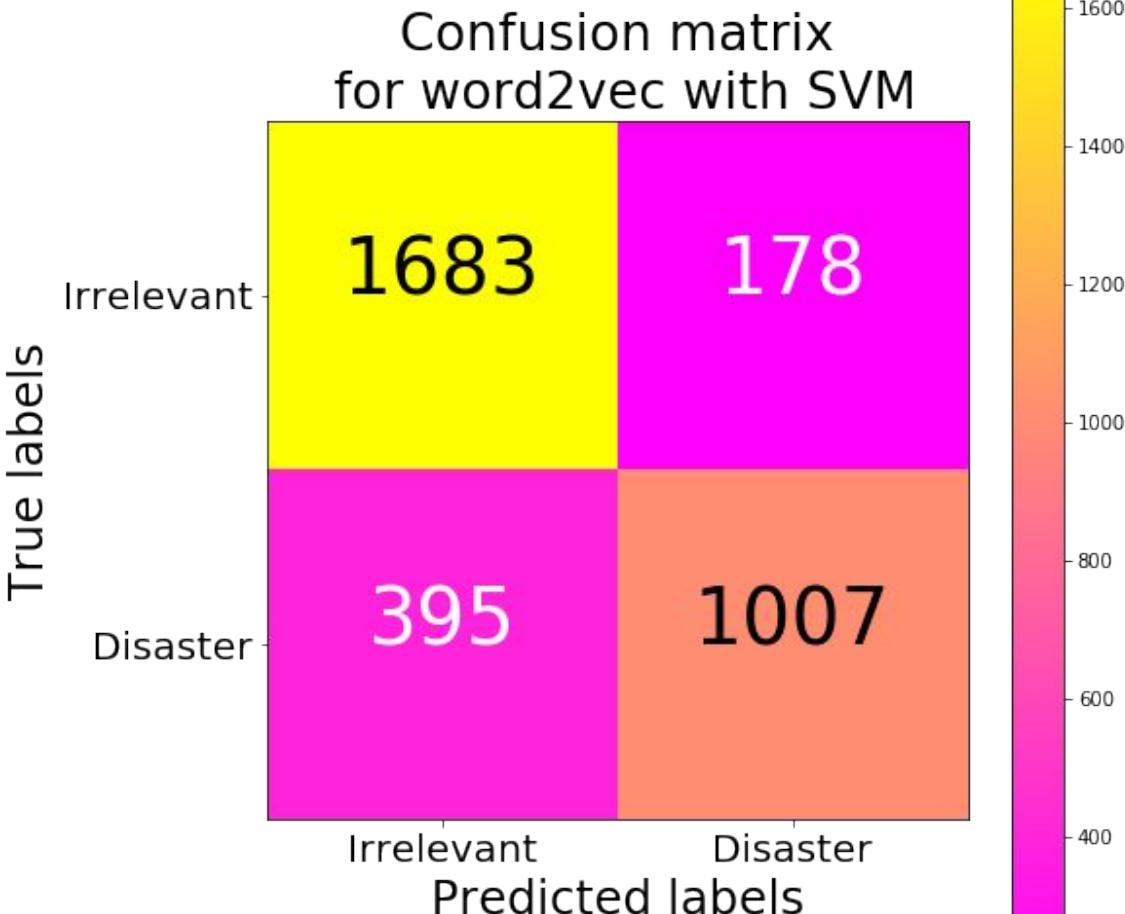
accuracy = 0.8443
f1 = 0.8426
precision = 0.8461
recall = 0.8443

TEST

accuracy = 0.8244
f1 = 0.8219
precision = 0.8270
recall = 0.8244

USED TO BE

accuracy = 0.8195
f1 = 0.8166
precision = 0.8227
recall = 0.8195



COMMIT HISTORY

SVM with Words2Vectors Mapping (version 17/17)

5 hours ago by Kiril Kuzmin

From "SVM with Words2Vectors Mapping" Script

submission.csv

9 hours ago by henryascend

BERT - RELU + SIGMOID ##### Comment for this version Basic Cleaning Adam: lr=6e-6 MAX_LEN = 160 validation_split=0.2, epochs=3, batch_size=16

submission.csv

2 days ago by henryascend

BERT text basic cleaning with lr 2e-6 4 epochs

submission.csv

3 days ago by henryascend

BERT - with text cleaning

SVM with Words2Vectors Mapping_from_Kiril (version 2/8)

4 days ago by Sumi

From "SVM with Words2Vectors Mapping_from_Kiril" Script

SVM with Words2Vectors Mapping_from_Kiril (version 1/8)

4 days ago by Sumi

From "SVM with Words2Vectors Mapping_from_Kiril" Script

0.81083

0.82617

0.82617

0.84253

0.80163

0.80674

	Trying Hyper-parameter on SVM Classifier	2m ago with no data sources
	SVM with Words2Vectors Mapping	5h ago with multiple data sources 0.81083
	P Clean1_GloVe1_LSTM1_L1L2_FM_e20_Mar	4h ago with multiple data sources
	Fork LSTM of Shiba_Inu_KerasLSTM#2	7m ago with multiple data sources 0.74948 gpu
	P Shiba_Innu of BERT_Henry	6h ago in Real or Not? NLP with Disaster Tweets · GPU ·
	P GloVe 300d Clean with LSTM_Mart	13h ago with multiple data sources gpu
	P GloVe Twitter Clean with LSTM_Mart	1d ago with multiple data sources · GPU · gpu
	P GloVe Twitter with LSTM_Mart	1d ago with multiple data sources · GPU · gpu
	LSTM with GloVe_Mart	2d ago with multiple data sources · GPU · gpu
	Folk_of_BERT_Henry	2d ago with multiple data sources · GPU ·
	P Fork1 of Shiba_Inu_KerasLSTM#2	2d ago with multiple data sources gpu
	SVM with Words2Vectors_Mart	2d ago with multiple data sources
	BERT_Henry	3d ago in Real or Not? NLP with Disaster T
	newbook	3d ago in Real or Not? NLP with Disaster T
	Not_That_Basic_Module_Pragma	21d ago in Real or Not? NLP with Disas

Best Submission						
Successful						
Versions						
Version 18	5 hours ago	SVM with ...	201.5s	0 B	+3	-1
Version 17	2 days ago	SVM with ...	190.5s	0 B	+4	-5
Version 16	2 days ago	SVM with ...	189.4s	0 B	0	-15
Version 15	2 days ago	SVM with ...	195.6s	0 B	+137	-432
Version 14	4 days ago	SVM with ...	190.5s	0 B	+139	-108
Version 13	4 days ago	SVM with ...	396.4s	0 B	+4	-2
Version 12	4 days ago	SVM with ...	249.8s	0 B	+57	-362
Version 11	4 days ago	SVM with ...	4.7s	0 B	0	0
Version 10	4 days ago	SVM with ...	5.7s	0 B	+463	-179
Version 9	5 days ago	SVM with ...	5.6s	0 B	+251	-18

	Improve Cleansing and GloVe Twitter with LSTM_Mart	1d ago	Sharing
In [1]: # This Python 3 environment comes with many helpful analytics libraries installed			
Versions			
Version 5	a day ago	Improve Cl...	665.1s 0 B +1572 -9
Version 4	a day ago	Improve Cl...	170.5s 0 B +53 -233
Version 3	2 days ago	Improve Cl...	381.9s 0 B +33 -5
Version 2	2 days ago	Improve Cl...	414.7s 0 B +236 -35
Version 1	4 days ago	SVM with ...	341.3s 0 B +9 -5
Forked from	4 days ago	SVM with ...	249.8s 801.74 KB 0 0

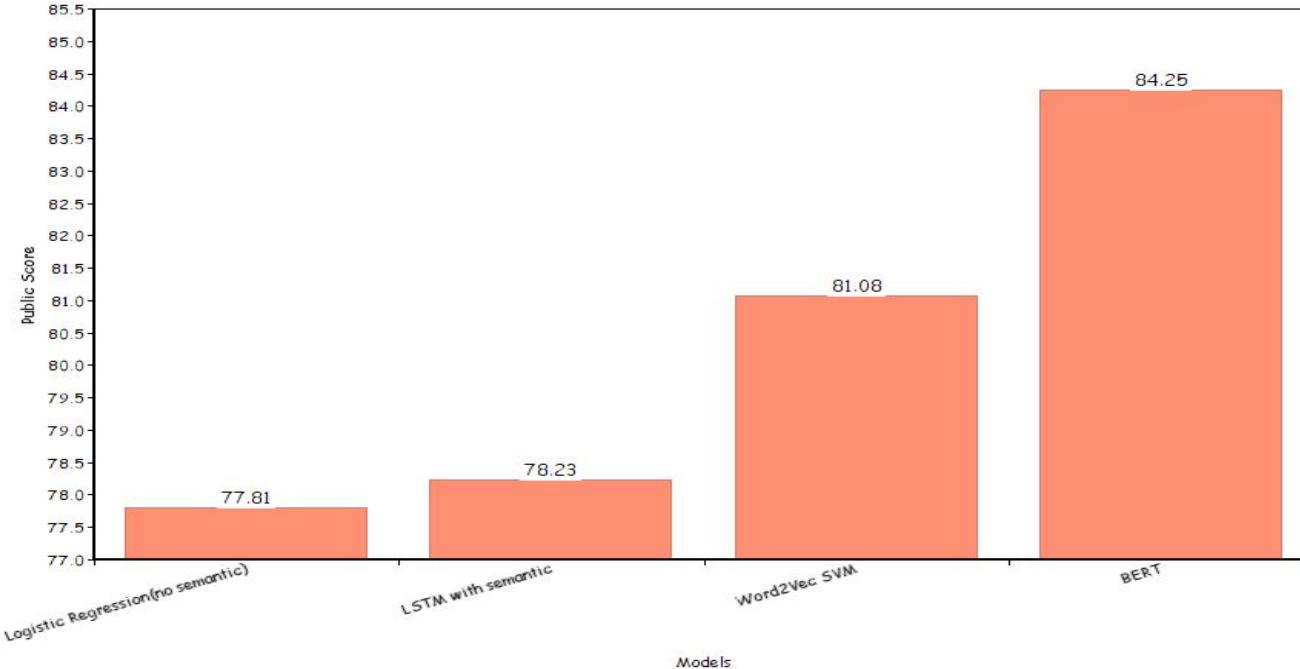
Remaining Meeting Time: 03:39



RESULTS

IMPLEMENTATION OF DIFFERENT MODELS

Implementation of different models

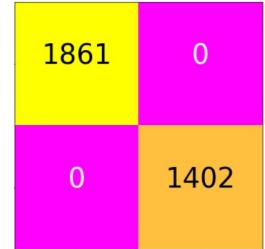
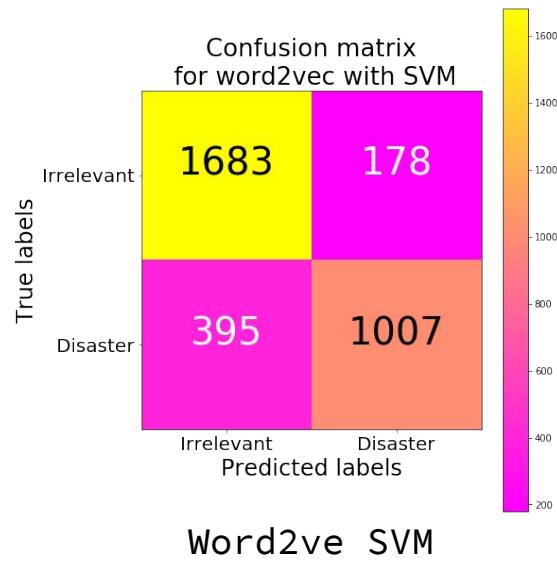
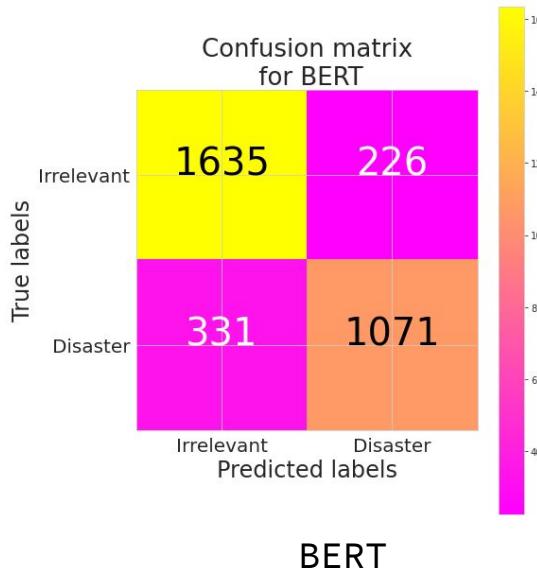
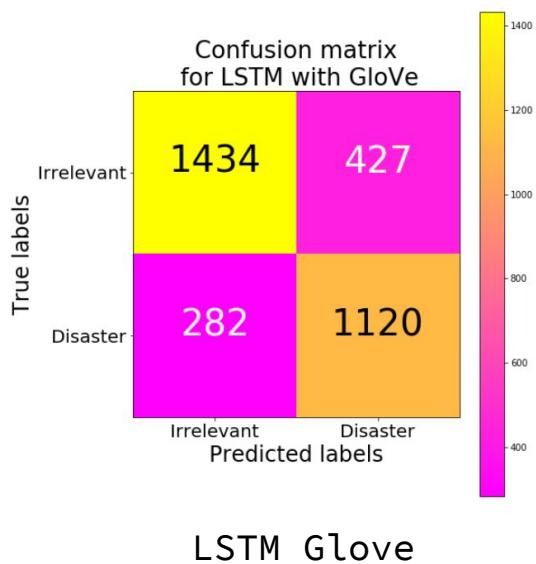


RESULTS: KAGGLE LEADERBOARD

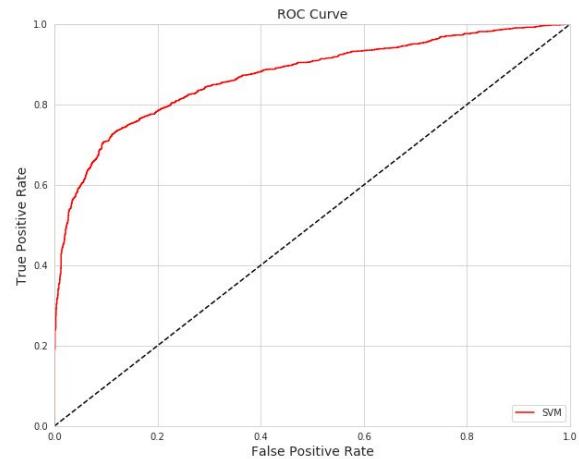
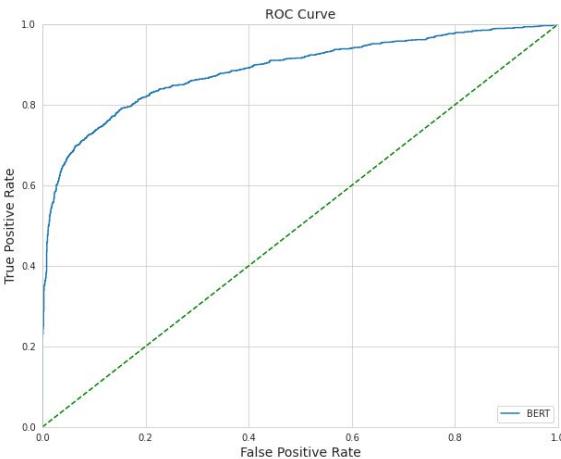
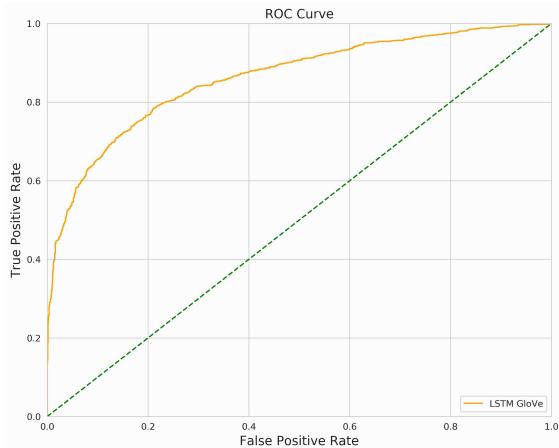
Overview	Data	Notebooks	Discussion	Leaderboard	Rules	Team	My Submissions	Submit Predictions
322	Chris X						0.84253	1 2mo
323	Nakul Agrawal						0.84253	11 1mo
324	Town Meng						0.84253	16 1mo
325	X.Y.Z						0.84253	9 20d
326	noob at best						0.84253	45 1mo
327	H.Eren						0.84253	19 1mo
328	Sudhir Kumar						0.84253	13 1mo
329	xuyiqi						0.84253	10 12h
330	Shiba_Inu						0.84253	25 4h
Your Best Entry ↑								
Your submission scored 0.82617, which is not an improvement of your best score. Keep trying!								
331	jeslyn&nuullll						0.84151	2 2mo
332	Gibran Otazo						0.84151	7 2mo
333	neptunexxx						0.84151	8 1mo
334	Shravan						0.84151	2 2mo
335	Ivtana						0.84151	16 2mo



CONFUSION MATRIX COMPARISON



ROC CURVE COMPARISON

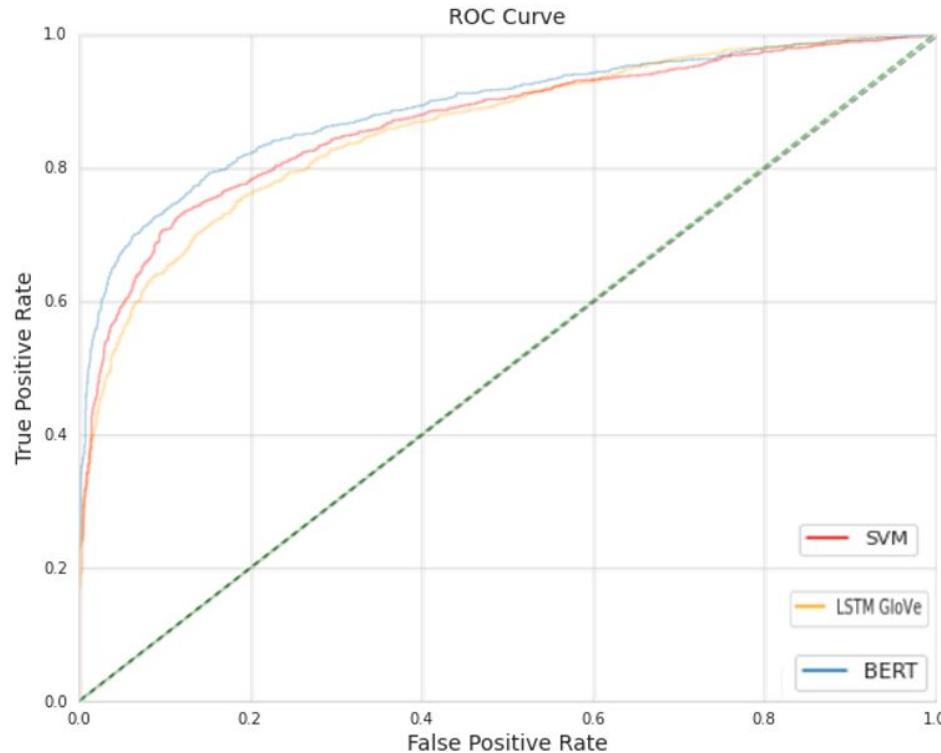


LSTM Glove

BERT

SVM

ROC CURVES TOGETHER



CONCLUSION

1. Some tweets have contradictory labels
2. BERT showed the best results:
 - a. public score: 0.8425
 - b. accuracy = 0.8774 f1 = 0.8474 precision = 0.8759 recall = 0.8276
3. Word2Vec is next:
 - a. public score: 0.8108
 - b. accuracy = 0.8244 f1 = 0.8219 precision = 0.8270 recall = 0.8244
4. Tomáš Mikolov is smart
5. Sometimes it can go off...
6. We have learnt:
 - a. how to use Kaggle platform
 - b. way to clean text data
 - c. NLP
 - d. shallow vs deep models for NLP
 - e. how to work in a team



[SVM with Words2Vectors Mapping](#) (version 15/18)

3 days ago by Kiril Kuzmin

0.00000

From "SVM with Words2Vectors Mapping" Script



THE END.

Q&A