

Beyond Raw Performance: Neural Machine Translation Based Multi-lingual Classification of Tweets for Automated Disease Surveillance

Mark Abraham Magumba (✉ magumbamark@hotmail.com)

Makerere University College of Computing and Information Sciences <https://orcid.org/0000-0002-2060-8625>

Peter Nabende

Makerere University College of Computing and Information Sciences

Research

Keywords: Data Mining, Epidemiology, Knowledge Engineering, Ontologies, Text classification

Posted Date: July 6th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-661600/v1>

License:  This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Beyond Raw Performance: Neural Machine Translation Based Multi-lingual Classification of Tweets for Automated Disease Surveillance

Mark Abraham Magumba^{1,2}, Peter Nabende^{1,3},

¹ Makerere University College of Computing and Information Sciences,
School of Computing and Informatics Technology, Department of Information Systems
magumbamark@hotmail.com, peter.nabende@gmail.com

Abstract. Twitter and social media as a whole have great potential as a source of disease surveillance data however the general messiness of tweets presents several challenges for standard information extraction methods. Most deployed systems employ approaches that rely on simple keyword matching and do not distinguish between relevant and irrelevant keyword mentions making them susceptible to false positives as a result of the fact that keyword volume can be influenced by several social phenomena that may be unrelated to disease occurrence. Furthermore, most solutions are intended for a single language and those meant for multilingual scenarios do not incorporate semantic context. In this paper we experimentally examine a translation based approach that allows for incorporation of semantic context in multi-lingual disease surveillance in the social web.

Keywords: Data Mining, Epidemiology, Knowledge Engineering, Ontologies, Text classification

1 Introduction

Disease surveillance methods based on Twitter surveillance typically count the volume of messages about a given disease topic as an indicator of actual disease activity via keywords such as the disease name [1- 3]. The most prominent use cases are nowcasting and forecasting. Nowcasting involves tracking outbreaks as they occur but due to time-lags in official reporting systems these systems can potentially “predict” outbreaks ahead of official systems by up to a few weeks [4]. Forecasting on the other hand involves longer time horizons [5] of up to several weeks. Generally speaking some positive correlation is assumed between the volume of messages and disease activity at a given time. However, in many cases this assumption is too strong as the volume of disease related messages can be influenced by panic and other factors. Therefore it is important to incorporate the semantic orientation of tweets to discriminate between relevant and irrelevant mentions of given keywords as in many cases even messages that explicitly mention diseases may actually do so in a non-occurrence related contexts or contexts that are spatio-temporally irrelevant. For instance the post, “I remember when the Challenger went down, I was home sick with

the flu!” is an actual reference to an occurrence of the flu but the Challenger disaster occurred in 1986 therefore it would be incorrect to count this mention to model an outbreak in 2018. There is some experimental evidence to suggest that incorporation of semantic orientation of tweets actually improves the end-to-end performance of prediction models for applications like nowcasting [6, 7]. In spite of this we are not currently aware of any large scale automated surveillance systems actively using semantic filtering techniques to classify messages. For multi-lingual applications the dominant approach has been use of multi-lingual ontologies and taxonomies the BioCaster [8] and HealthMap [9] systems. These are still essentially keyword volume systems at the core. Furthermore, these systems are really just secondary aggregators as they themselves rely on human mediated systems like Promed-Mail [10] which may make the semantic check redundant.

In addition, social media messages generated on platform like Twitter present unique challenges for conventional text processing techniques for instance Twitter text is generated by millions of users, each with their own individual writing style and vocabulary. In addition tweets are short, colloquial in nature and are characterized by slang, misspellings, poor grammar and additional artifacts like hashtags, emoticons and URLs (Uniform Resource locators). Regarding multi-lingual message classification, there are two possible approaches. The first entails creating different models for each language under consideration. This is potentially resource intensive as it requires multi-lingual expertise. The second is to create a single model in a so called “resource-rich” language and then employ it to classify related “resource-poor” languages. This requires the “resource-poor” languages to be translated to the “resource-rich” language. Given good translation can be obtained; this approach makes sense and should produce good results. Unfortunately, the same problems that make multilingual message classification difficult for Twitter messages also make state of the art message translation suboptimal. However, these state of the art data driven machine learning translation systems such as Google Translate¹ and Bing Microsoft Translator² are becoming better with time and incorporating an increasing number of languages. For instance as of June 2021 Google Translate supported 109 languages and Bing Microsoft Translator supported 90 languages.

The state of the art machine translation systems employ Neural machine translation approaches which rely on recurrent auto encoder – decoder neural networks. It also helps to have in domain training data such that the style and vocabulary of the source data is similar to the target as words may have different meanings in different situations. As input, these systems take distributed representations of the source language and output distributed representations of the target language. In this work we employ Google’s Neural Machine Translation system (GNMT). The GNMT models are trained using publicly available data from sources like European Union communications. These are generally formal in style as opposed to the colloquial style of Twitter messages and as a consequence the translation performance of systems like GNMT is unreliable on tweets. However, as already stated it is improving and in

¹ <https://translate.google.com>

² <https://bing.com/translator>

some cases like GNMT freely available. To deal with rare words, GNMT employs word pieces as the lexical unit rather than actual words. This is quite useful for Twitter as it makes GNMT somewhat robust to certain types of errors like slight misspellings and enables better handling of rare words.

2.0. Materials and Methods

We can summarise the steps taken for our experiments as an activity pipeline. The pipeline is summarised in figure 1.

[insert fig1]

Figure 1: Multi-lingual Message Classification Pipeline

2.1. Corpus generation:

The first step is the creation of the corpus. We obtain tweets from a basic Twitter account using some specific keywords via a python script through Twitter's Streaming API using the python tweepy plugin³. The tweets we download are those that are marked as public which is the default security level and they are only marked private if expressly indicated by users. We employ simple keyword filters to extract the desired tweets. For the training data we employ a data set of 13004 English tweets that mention the flu, common cold or Listeria.

For the test data we extract tweets that mention the flu in French, German, Spanish, Arabic and Japanese. We translate the tweets eliminating those tweets that are incomprehensible and then annotate the remainder of the tweets. At annotation we label those tweets that mention a recent (less than a month) or ongoing cases of disease as positive. Within this period we expect the disease to still be within its communicable period which is a period within which new infections are still possible as a result of transmission from sick individuals to susceptible healthy individuals. We treat all other mentions as irrelevant and we label them messages negative.

In addition we eliminate duplicates by removing retweets, (tweets with the “RT” tag) and also manually check for duplicates that may not be marked as “RT” and finally we remove all punctuation except the “#” and “@” symbols where they appear at the beginning of tokens where they are used to denote hashtags and users respectively. Table 1 below summarizes the composition of our corpus. We do not perform any preprocessing prior to translation. We found the translation API to be quite robust but in many cases it returns some unspecified system error and in a few cases the translation only contains inconsequential elements like URLs. For this reason there are less tweets retained after translation than the actual number of tweets

³ <https://github.com/tweepy/tweepy>

contained in the corresponding datasets. The column for yield in table is the percentage of tweets successfully translated and gives some impression of the performance of the translation API on tweets from different languages.

Table 1: CORPUS SUMMARY

Language	#Speakers (millions)	#Territories spoken	Language family	#Tweets	#Retained	Yield	Split	
							% Positive	% Negative
English	510	> 50	Indo-European	13004	-	-	0.57	0.43
French	270	>30	Indo-European	510	401	0.78	0.33	0.67
German	220	>10	Indo-European	547	400	0.73	0.36	0.64
Spanish	420	>20	Indo-European	523	340	0.65	0.67	0.33
Arabic	255	>30	Afro-Asiatic	503	288	0.57	0.30	0.70
Japanese	127	1	Japonic	553	400	0.72	0.28	0.72

2.2. Pre-processing:

The next step is pre-processing. We start off by tokenization then part of speech tagging. For part of speech tagging we employ the GATE (General Architecture for Text Engineering) Twitie tagger application [11]. The tagger uses the Penn Treebank tag set [12] in addition to three additional tags “HT”, “USR” and “URL” corresponding to twitter specific phenomenon namely hashtags, users and URLs respectively. We also attempt these experiments with stemming and without stemming. Stemming further reduces lexical diversity by reducing different forms of the same word into a single stem.

2.3. Feature Generation

We try out four different input representations. We try word one-hot word-level representations, one-hot concept-level representations, distributed representations over words and distributed representations over concepts. For the conceptual representations we employ two different ontologies. We try the ontology previously developed by Magumba et al [13] and SNOMED-CT (Systematic Nomenclature of Medicine-Clinical Terms) [14]. Whereas several ontologies exist such as OBO (Open Biomedical Ontologies) [15] and the BioCaster ontology [8], we elect to employ these two because they have a broader conceptual coverage which we consider an advantage in a general language modelling task. For instance the BioCaster method partially defined word lists that contain terms such as “human ehrlichiosis” and “enzootic bovine leukosis” which are generally too technical to appear in casual texts

like Twitter posts with any significant regularity. The Magumba et al [13] ontology on the other hand was created specifically for Twitter disease event detection and SNOMED-CT was created specifically to harmonise divergent medical terminology and is marketed by SNOMED-CT international as the most comprehensive and precise clinical health terminology product in the world.

For the ontology based input representations we transform each tweet into a vector of features as follows: Firstly, we flatten out our ontology into a list of its constituent concepts. For the Magumba et al [13] ontology each concept is associated with a group of words or tokens referred to as the concept dictionary. Each concept is effectively a list of words and the full ontology is basically a list of lists. In this sense it is a heavily redacted English dictionary containing only words considered to be of epidemiological relevance. To obtain the feature vector we simply tokenize each tweet and for each token we do a dictionary look up in the flattened ontology. If the token exists in the ontology, we simply replace it with the concept in which it occurs. As an example the sentence “I have never had the flu” is encoded as “SELF_REF HAVE FREQUENCY HAVE OOV OOV”.

SELF_REF refers to “Self references” which is the concept class for terms that persons use to refer to themselves such as “I”, “We” and “Us” used as an indicators of speaking in the first person, “HAVE” is the concept class for “have” or “had” which is a special concept class since the verb “to have” is conceptually ambiguous as it can legitimately indicate two senses that is falling sick or possession. The “FREQUENCY” terms refers to a reference to frequency concept which denotes temporal periodicity. The “OOV” terms at the end of the CNF representation stands for “Out of Vocabulary”. The current version of this ontology has 136 concepts corresponding to 1531 tokens versus a vocabulary of about 59,000 tokens for our full corpus (or several billion words in English). Needless to say, most words are out of vocabulary. To obtain the final so-called CNF representation the “OOV” terms are replaced with their part of speech tag therefore our previous example, “I have never had the flu” becomes “SELF_REF HAVE FREQUENCY HAVE DT NN”. Figure 2 below depicts the transformations for the message “I have never had the flu!”.

For SNOMED-CT the ontology is organized differently, each concept has a corresponding description but there is no concept of a concept dictionary. For the SNOMED-CT experiments we employ an SQLite3 implementation, for each word we search the concept for which it appears in the description using the FTS4 (Full-text Search) engine. We deal with out of vocabulary concepts the same way by replacing them with their part of speech tags. For both ontology representations the input may take the form of a simple one-hot vector or as a distributed representation by applying neural embeddings over concepts in the same way they are applied to words.

[insert fig2]

Figure 2: Deriving Feature Vector from CNF and POS Tags

2.4. Word2vec/ Doc2Vec Settings

For training distributed embeddings we use the word2vec/doc2vec model by Mikolov et al [16, 17]. For the ontology-based CNF word2vec/doc2vec model we find optimal performance with a 200 dimensional representation, 8 noise words, a context window of 5, and 20 training epochs with distributed memory architecture and minimum word count of 2. For the word-level word2vec model we employ Google’s 300 dimensional gold standard 3 million word corpus.

2.5. Experimental Setup

We try out a variety of models including deep neural networks using Convolutional Neural networks (CNNs) and Recurrent Neural Networks (RNNs) with Long Short Term Memory (LSTM) units. We specify a maximum message length of 20 tokens for these experiments. Consequently, each tweet takes the form of a 20 X 200 vector representation, messages that are shorter than 20 words are zero padded. For the CNN-CNF model we use a similar architecture to Yoon Kim [18]. We first pass the embedding to a dropout layer then employ three filters of width 3, 5, 7 and a length the same as the concept embedding length of 200 rectified linear units (ReLus) and a stride of 1. The output from the feature maps is passed through a max-pooling layer and their full output concatenated into a final feature vector that is fed to a dropout layer which then feeds into a sigmoid output layer with one neuron. The model architecture is depicted in figure 3. The architecture is similar for the word-level experiment that employs Google’s gold standard model except since it is a 300 dimensional representation, the input shape is 20 X 300 instead.

[insert fig3]

Figure 3: Model Architecture for CNN Classification Model

For the CNN-LSTM-CNF model, which stacks an RNN on top of a CNN, we employ a single filter of width 3 and a stride of 1, then apply max-pooling with a pool width of 2 to the resulting feature maps. We then concatenate the output into a single feature vector which we pass into an LSTM layer that feeds directly to a sigmoid output layer containing one neuron. Figure 4 below depicts the model architecture.

[insert fig4]

Figure 4: Model Architecture for CNN-LSTM Classification Model

For the Stack of two LSTMs-CNF model, which stacks an RNN layer on top of another, we first apply dropout to the input array then feed the output to an LSTM layer 200 neurons wide for the CNF model followed by a dropout layer then another

LSTM layer followed by another dropout layer which finally feeds into a sigmoid output layer with a single neuron. Figure 5 depicts the model architectures the stack of 2 LSTMs model for the input message “I THINK AM GOING TO GET SICK” in the CNF form described by Magumba et al [13].

[insert fig5]

Figure 5: Model Architecture for Stack of 2 LSTMs classification Model

For the Bi-directional LSTM-CNF model we first apply dropout to the input layer then feed the result into two parallel stacks of LSTM layers. The output is ordered front to back in the second stack. Each stack comprises a first layer 200 neurons wide whose output is fed to a dropout layer which then feeds its output to a second LSTM layer. At this point the output from both stacks is concatenated and fed into a single dropout layer and into a final sigmoid output layer with a single neuron. Figure 6 below depicts the Bi-directional LSTM model for the input sentence, “SICK WITH THE FLU!” For all deep learning experiments we find the optimal performance after a small number of iterations. We use five epochs for model training, beyond this the models tend to overfit due to the small size of the effective vocabulary. All coding is done in python and for the neural models we employ the python keras package. For all CNN and LSTM models the dropout layers are with a dropout probability of 0.25.

[insert fig6]

Figure 6: Model Architecture for Bi-Directional LSTM Classification Model

For word2vec and Doc2Vec vectors we employ the python gensim package [19]. For the unigram bag of words models we use scikit-learn’s SGD classifier which is a support vector machine classifier trained with a stochastic gradient descent procedure. We also employ the python scikit- learn package [20] for the logistic regression classifier. For the logistic regression and SGD models we use the scikit-learn hyper parameter defaults except we employ 10,000 iterations for the SGD model.

3.0. Results and Discussion

We employ the precision, recall and F1 Score as our base performance metrics. They are given by the following equations:

$$P = \frac{TP}{TP + FP} \quad (1)$$

$$R = \frac{TP}{TP+FN} \quad (2)$$

$$F1 Score = \frac{2*P*R}{P+R} \quad (3)$$

Where P, R, TP, FP, FN represent Precision, Recall, True Positives, False Positives and False Negatives respectively

The results are presented in table 2 below. For clarity we separate the results into two parts, the first discusses one hot vector encoded representations whilst the second discusses distributed neural embeddings. To measure the model performance we employ un-weighted average performance in addition to the performance. The ideal situation is to have a high average performance coupled with a small performance variance. A small variance means the classifier's performance on different datasets is stable, a high variance means the performance characteristics of the approach vary greatly from language to language. To combine the two into a single measure we use the following formula:

$$Overall\ performance = 2 * \frac{[1-Norm(Var)]*Avg(Performance)}{[1-Norm(Var)]+Avg(Performance)} \quad (4)$$

We refer to the quantity 1- Norm(Var) as the “invariance”. The quantity Norm(Var) is the “normalized variance” which is the performance variance expressed as a percentage of the maximal possible variance. Since the base performance metrics (precision, recall and F1Score) are bounded between 0 and 1, it means the variance also has an upper bound which can be calculated from Popovici's inequality as

$$\sigma^2 \leq \frac{M-m}{4} \quad (5)$$

Where M is the largest possible value and m is the smallest possible value. In our case M = 1, and m = 0.

This ensure that the invariance ranges from 0 to 1 and our overall performance also ranges from 0 to 1. Therefore an overall score of 0 implies that the performance of the model for some metric is 0 for all datasets or the performance variance is maximal for instance where precisely half of the datasets have a maximal score for the metric and the other half have the minimal score. An overall score of 1 on the other hand, for a given metric, implies that the model obtains a perfect score for the given metric in all of the language datasets.

We have also excluded the results of the validation dataset from the overall model performance as conveyed in the aggregate score columns (last three columns), their

inclusion does not provide any additional information since it does not change the order of aggregate model performance. The first row indicates the performance with the unigram bag of words baseline whilst the remaining rows indicate the performance of different approaches to mitigating performance divergence that may occur between the training data and real world data mainly due to cross-lingual lexical divergence and sub-optimal translation.

We find only modest variations between the best performing approaches for different representations (indicated in bold). The highest average performance for categorical one hot vector representations with the bigram SNOMED-CT model with an un-weighted average performance of 0.59, 0.66 and 0.62 respectively for precision, recall and F1 score respectively. The overall scores are 0.72, 0.77 and 0.75 for precision, recall and f1 score respectively. The best performing distributional model is the CNN + word2vec model with an un-weighted average precision, recall and F1 score of 0.55, 0.72 and 0.63 respectively and an overall performance of 0.69, 0.81 and 0.75 for precision, recall and F1 score respectively. Crucially, their overall F1 score performance is tied at 0.75. In fact, across methods it is seen that there are only very slight differences between the performance of different methods and representations. For instance both best performing approaches are only 1.3% better than the CNF model in terms of overall performance by F1 score. In addition they are only 4.2% better than the baseline model in terms of overall performance by F1 score. So, there isn't really a huge performance pay-off for the additional technical complexity of using conceptual representations and deep neural approaches. We also score consistently higher recall than precision across all methods.

As to why we notice very slight differences between different representations and very slight gains in performance versus the unigram word-level baseline, it may be because we employ messages about the same disease therefore resulting in smaller lexical differences implying the problem these approaches are designed to solve didn't exist in the data as selected.

It's also noteworthy that the good performance of the SNOMED-CT model is a bit counterintuitive. The reason for this is that SNOMED-CT is not organized for this sort of task, for instance we employ an SQLite3 implementation and in order to obtain the concept referred to by a token we rely on a full text search via the FTS4 (Full text Search) engine⁴. SNOMED places these terms in the description of the concept, as an example a search for the terms "Man" and "Woman" would produce the following truncated output as depicted in figure 7 and 8 respectively:

[insert fig7]

Figure 7: Truncated output from SNOMED-CT search for "Man"

⁴ <https://www.sqlite.org/fts3.html>

[insert fig8]

Figure 8: Truncated output from SNOMED-CT search for “Woman”

As can be seen from figures 7 and 8, several candidate concepts are returned. We would prefer for both cases to map to the “person” concept but there is no way of automatically enforcing this in our setup. In our case we simply return the first match which means that man and woman get mapped to different concepts. Moreover these aren’t the preferred meanings as “Man” and “Woman” return “Stiff-man syndrome” and “Achard-Thiers syndrome” respectively which are disorders. From a text processing point view this is no better than a word level model as both words are mapped to different atoms. For the model, having encountered “Man” yields no useful information on “Woman”. The good performance of the SNOMED-CT based conceptual representation relative to the baseline is hard to explain given this, it could be a result of the fact that it has a very wide coverage and even though mappings are frequently meaningless they are consistent enough that it somewhat effectively normalizes the text since most words are mapped to some concept (even though it may be the wrong one) and there are fewer out of vocabulary terms than the CNF representation. The SNOMED-CT encoder returns a result 98.28% of the time versus 90% of the time for the CNF encoder therefore it effectively acts as a sort of text normalizer.

Finally, it is evident that we obtain poorer performance on languages which are more distant from the model language as results are consistently poorer on the Japanese and Arabic datasets with Japanese and Arabic being from the Japonic and Afro-Asiatic language families as opposed to French, German and Spanish which like English are from the Indo-European family.

Table 2: Results of experiments for different methods, representations and data sets

Methods	Measure	Datasets						Aggregated Scores		
		English (Validation dataset)	French	German	Spanish	Arabic	Japanese	Un-weighted Average Performance	Performance Variance	Overall Performance
Categorical One Hot vector Representation										
Unigram BOW (Baseline)	Precision	0.8435	0.58	0.56	0.81	0.42	0.48	0.57	0.02	0.70
	Recall	0.8144	0.74	0.8	0.81	0.72	0.29	0.67	0.05	0.74
	F1 Score	0.8747	0.65	0.66	0.81	0.53	0.36	0.6	0.03	0.72
Unigram BOW (Stemmed)	Precision	0.8082	0.51	0.54	0.75	0.42	0.43	0.53	1.30E-02	0.68
	Recall	0.7695	0.77	0.82	0.86	0.79	0.34	0.72	3.60E-02	0.78
	F1 Score	0.8508	0.61	0.65	0.8	0.55	0.38	0.6	1.80E-02	0.73
Unigram BOW CNF	Precision	0.7278	0.53	0.5	0.76	0.44	0.39	0.52	0.02	0.67
	Recall	0.7387	0.79	0.68	0.87	0.73	0.59	0.73	0.01	0.83
	F1 Score	0.7173	0.63	0.58	0.81	0.56	0.47	0.61	0.02	0.74
Uni/Bigrams + CNF	Precision	0.8555	0.55	0.5	0.79	0.5	0.41	0.55	1.60E-02	0.69
	Recall	0.8197	0.77	0.67	0.79	0.69	0.51	0.68	9.50E-03	0.80
	F1 Score	0.8947	0.64	0.57	0.79	0.58	0.46	0.61	1.10E-02	0.74
Unigram Snomed	Precision	0.813	0.59	0.55	0.78	0.47	0.44	0.57	1.40E-02	0.71
	Recall	0.789	0.78	0.77	0.82	0.6	0.4	0.67	2.40E-02	0.77
	F1 Score	0.8385	0.67	0.64	0.8	0.53	0.42	0.61	1.60E-02	0.74
Bigram Snomed	Precision	0.8197	0.58	0.56	0.79	0.53	0.47	0.59	1.20E-02	0.72
	Recall	0.8947	0.73	0.74	0.83	0.6	0.41	0.66	2.00E-02	0.77
	F1 Score	0.8555	0.65	0.64	0.81	0.56	0.44	0.62	1.40E-02	0.75
Distributed Neural Representations										
CNN + word2vec	Precision	0.7579	0.55	0.53	0.78	0.4	0.48	0.55	1.60E-02	0.69
	Recall	0.7336	0.84	0.77	0.9	0.6	0.51	0.72	2.10E-02	0.81
	F1 Score	0.7839	0.67	0.63	0.84	0.51	0.5	0.63	1.50E-02	0.75
CNF LSTM stack	Precision	0.7229	0.55	0.51	0.76	0.49	0.33	0.53	0.02	0.67
	Recall	0.7055	0.76	0.62	0.87	0.67	0.46	0.68	0.02	0.77
	F1 Score	0.8947	0.64	0.56	0.81	0.56	0.38	0.59	0.02	0.71
CNN CNF	Precision	0.7227	0.48	0.46	0.73	0.36	0.29	0.46	0.03	0.61
	Recall	0.5656	0.86	0.87	0.94	0.76	0.7	0.83	0.01	0.89
	F1 Score	1	0.61	0.6	0.82	0.49	0.41	0.59	0.02	0.71
Bi LSTM CNF	Precision	0.8063	0.57	0.54	0.76	0.5	0.33	0.54	0.02	0.68
	Recall	0.7393	0.72	0.67	0.77	0.63	0.51	0.66	0.01	0.78
	F1 Score	0.8864	0.64	0.6	0.76	0.56	0.4	0.59	0.02	0.72
CNN-LSTM CNF	Precision	0.7500	0.55	0.47	0.79	0.37	0.37	0.51	0.03	0.64
	Recall	0.7500	0.83	0.65	0.8	0.61	0.61	0.7	0.01	0.81
	F1 Score	0.7500	0.67	0.55	0.79	0.46	0.46	0.59	0.02	0.72

4.0. Conclusion and Future Work

The results are promising particularly for languages closely related to the model language. The performance is significantly stronger on languages in the Indo-European language family to which English, the model language, belongs. This is probably due to an abundance of parallel datasets for model development for closely related languages than more distant languages like Arabic. For a globally deployed live implementation a divide and rule strategy could be applied in which different models are created for different groups of languages. Still this would be far cheaper than creating language specific models for the thousands of languages that exist.

To get a mathematical impression of the significance of this work we can take the example of a nowcasting application. As already stated, nowcasting approaches that rely on textual web data typically model disease intensity as a function of keyword volume. If we consider the Spanish dataset for which we obtain the highest performance and 67% of tweets are positive, a nowcasting model would have an input error of 50% without message classification. That is if the nowcasting model assumed that all messages containing the keyword “flu” were relevant; the model would incorrectly assume disease activity to be 50% more intense than it actually is. Ideally the model should only accept those messages which report actual occurrences of disease, in actuality it will accept any messages labeled as relevant. The ideal and actual model inputs and model input error are given by the expressions below:

$$\text{Ideal Model Input} = TP + FN \quad (6)$$

$$\text{Actual model input} = TP + FP \quad (7)$$

$$\text{Model input error} = FP - FN \quad (8)$$

Where TP, FP, FN represent True Positives, False Positives and False Negatives respectively

Any precision performance less than 1, means that false positives exist which would increase the model input error, any recall performance less than 1 means false negatives exist which reduces the model input error. In general a negative model error implies that the precision is better than the recall and a positive error that the recall is better than the precision. Crucially, by combining equations 8 with equation 1 and 2 we get the following expression equivalence for the error rate.

$$\begin{aligned} \text{Error rate} &= \frac{\text{Model input error}}{\text{Ideal Model Input}} = \frac{FP - FN}{TP + FN} = \frac{\left[\frac{R - P}{PR} \right] * TP}{\frac{TP}{R}} \\ &= \frac{R - P}{P} \end{aligned}$$

Where P, R, TP, FP, FN represent Precision, Recall, True Positives, False Positives and False Negatives respectively

Therefore the error rate is proportional to the difference between the recall and precision. This is because any false negatives are offset by false positives. Therefore, if the precision and recall are equal as with the Spanish dataset using the unigram/bigram CNF model (in bold italics) then the assuming these errors are not geo-spatially localized (for example if they are a product of phenomena like dialect differences) the error would effectively be zero.

Therefore for the end to end surveillance performance, it isn't even necessary to have a perfect classifier at the linguistic step but rather one which has a balanced performance in terms of precision and recall. In this respect the bigram-SNOMED model is doubly desirable as it not only has the highest overall performance but also the least difference in recall and precision performance overall. Where a balanced precision and recall performance is not achievable a higher precision performance is preferred to avoid false positives and ultimately reliable performance (i.e. performance with a low variance as data changes) is preferred to avoid unreliable results. The raw performance may not be that important as long as the performance is predictable and can therefore be adjusted for in the surveillance models.

Declarations

Competing Interests

None

Authors' Contribution

Authors' contributions MAM conceptualized the idea, prepared the datasets, designed and run the experiments, analyzed results, drafted and proof-read the manuscript, PN provided general guidance concerning the work flow of the solution, verified datasets, proof-read and revised initial versions of the manuscript, verified the results. All authors read and approved the final manuscript.

Author details

¹ Department of Information Systems, Makerere University College of Computing and Information Sciences, Kampala, Uganda

Acknowledgements

We would like to acknowledge SNOMED-CT international for giving us free access to their ontology

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The datasets used in these experiments including the best performing Doc2Vec model file can be found in compresses form at our Google drive folder, Google's 6 billion word corpus word2vec model can be downloaded at <https://drive.google.com/file/d/0B7XkCwpI5KDYNINUTTlSS21pQmM/edit>.

Consent for publication

We, the authors, consent to the publication of this manuscript in the Journal of Big Data.

Ethics approval and consent to participate

Not applicable.

Funding

Parts of this research were funded by the Norwegian Agency for Development Cooperation – Health Informatics Training and Research in East Africa for Improved health Care (NORAD – HITRAIN) project.

References

1. Lee K, Agrawal A, Choudhary A. Real-time disease surveillance using twitter data: demonstration on flu and cancer. Paper presented at: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013.
2. Paul MJ, Dredze M. Discovering health topics in social media using topic models. PloS one. 2014;9:e103408.
3. Souza RC, Assunc˜ao RM, de Oliveira DM, de Brito DE, MeiraJr W. Infection Hot Spot Mining from Social Media Trajectories. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2016. p. 739–755.
4. Eiji A, Sachiko M, Mizuki M. Twitter Catches the Flu: Detecting Influenza Epidemics Using Twitter. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011.p. 1568–1576. Available from: <http://dl.acm.org/citation.cfm?id=2145432.2145600>.
5. Beswick A. # Outbreak: An Exploration of Twitter metadataas a means to supplement influenza surveillance in Canada during the 2013–2014 influenza season, 2016
6. Doan S, Ohno-Machado L, Collier N. Enhancing Twitter data analysis with simple semantic filtering: Example in tracking influenza-like illnesses. Paper presented at: Healthcare Informatics, Imaging and Systems Biology (HISB), 2012 IEEE Second International Conference on, 2012.
7. Lamb A, Paul MJ, Dredze M. Separating fact from fear: Tracking flu infections on twitter. Paper presented at: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013.
8. Collier N, Doan S, Kawazoe A, et al. BioCaster: detecting public health rumors with a Web-based text mining system. Bioinformatics. 2008;24:2940–2941.
9. Chen H., Zeng D., Yan P. (2010) HealthMap. In: Infectious Disease Informatics. Integrated Series in Information Systems, vol 21. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-1278-7_14

- 10.Brownstein JS, Freifeld CC, Madoff LC. Digital disease detection – harnessing the Web for public health surveillance. *The New England journal of medicine*. 2009 May 21; 360(2):2153
- 11.Cunningham H, Maynard D, Bontcheva K, Tablan V. GATE: an architecture for development of robust NLP applications. Paper presented at: Proceedings of the 40th annual meeting on association for computational linguistics, 2002.
- 12.Taylor A, Marcus M, Santorini B. The Penn Treebank: an overview. In: Abeillé A, editor. *Treebanks., Building and using parsed corpora*Dordrecht: Springer; 2003. p. 5–22.
- 13.Magumba MA, Nabende P. An Ontology for Generalized Disease Occurrence Detection on Twitter. Paper presented at: International Conference on Hybrid Artificial Intelligence Systems, 2017.
- 14.Donnelly K. SNOMED-CT: The advanced terminology and coding for eHealth. *Studies in health technology and informatics* 2006 Jan; 121:279Smith B, Ashburner M, Rosse C, et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*. 2007; 25:1251.
- 15.Smith B, Ashburner M, Rosse C, et al. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*. 2007; 25:1251.
- 16.Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013.
- 17.Le Q, Mikolov T. Distributed representations of sentences and documents. Paper presented at: International Conference on Machine Learning, 2014.
- 18.Kim Y. Convolutional neural networks for sentence classification. arXiv preprint arXiv:14085882. 2014;
- 19.Rehurek R, Sojka P. Software framework for topic modelling with large corpora. Paper presented at: In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, 2010.
- 20.Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*. 2011;12:2825-2830.7Appendix: Springer-Author Discount

Figures

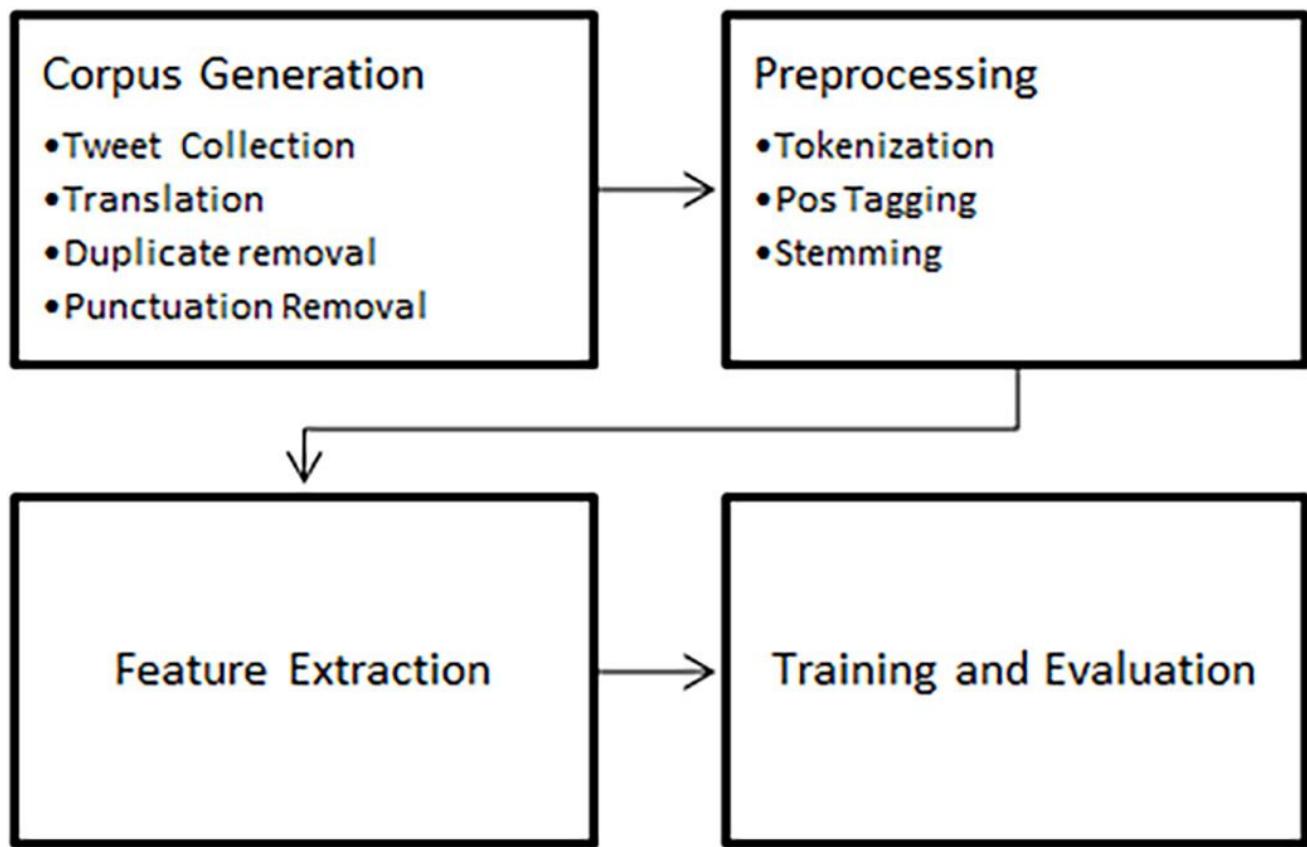


Figure 1

Multi-lingual Message Classification Pipeline

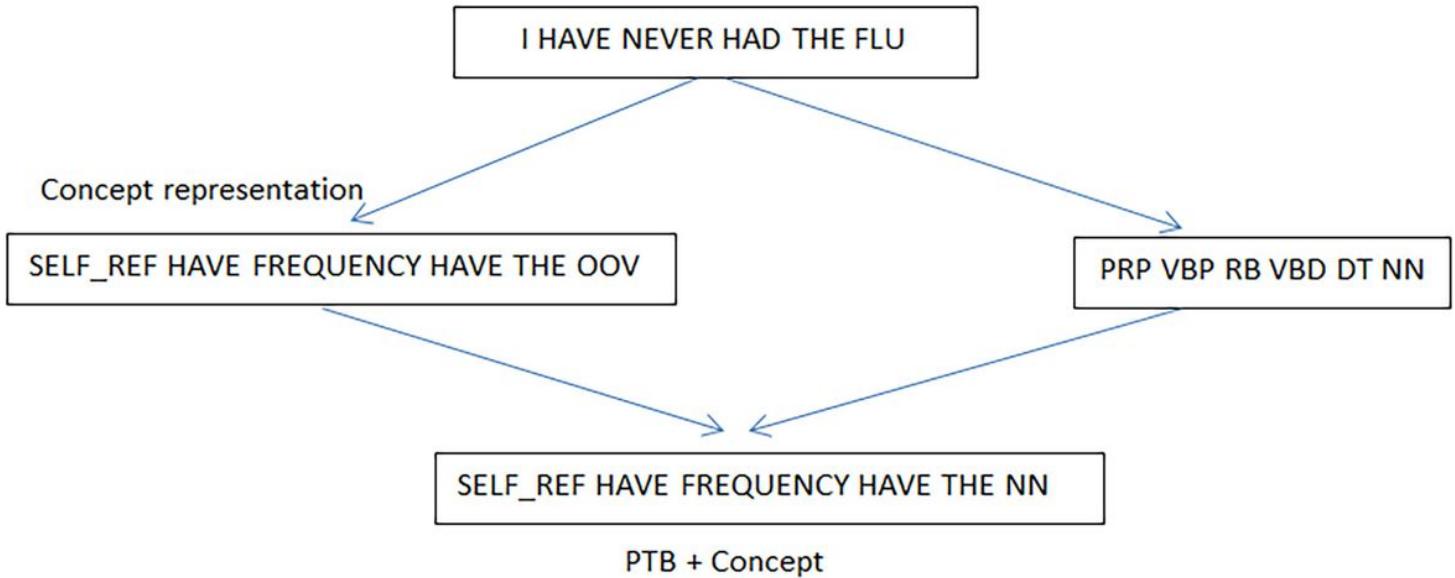


Figure 2

Deriving Feature Vector from CNF and POS Tags

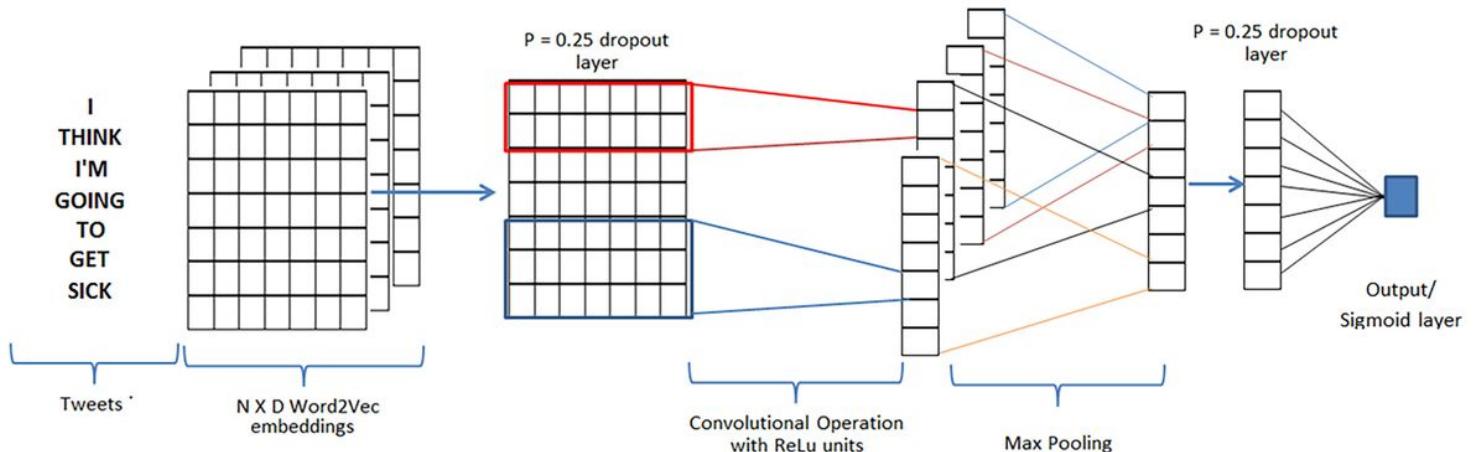


Figure 3

Model Architecture for CNN Classification Model

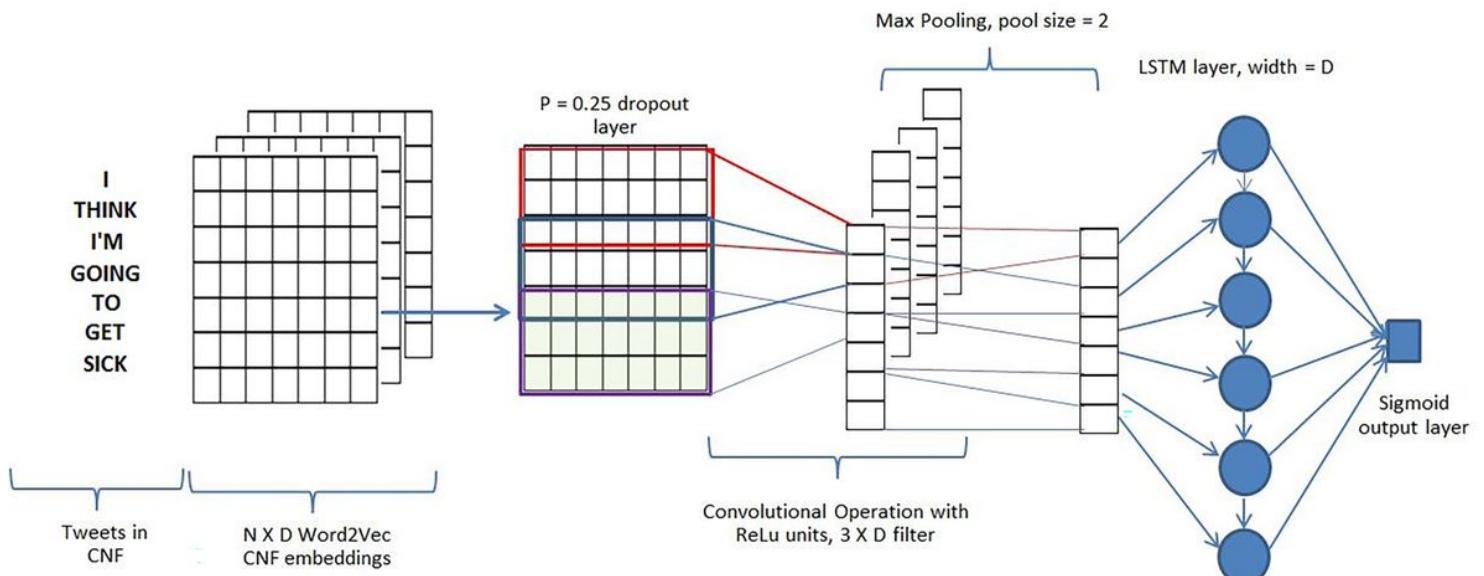


Figure 4

Model Architecture for CNN-LSTM Classification Model

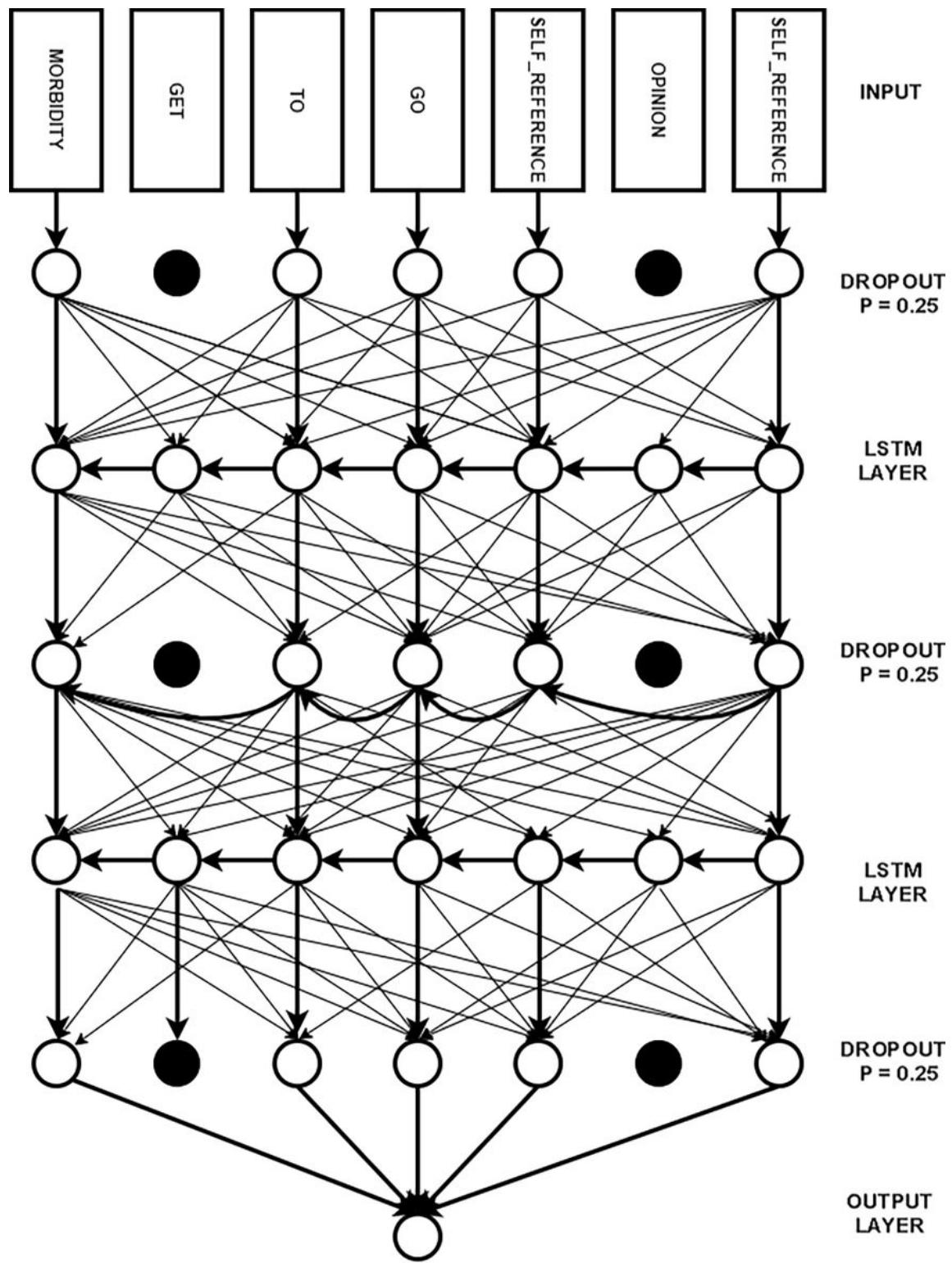
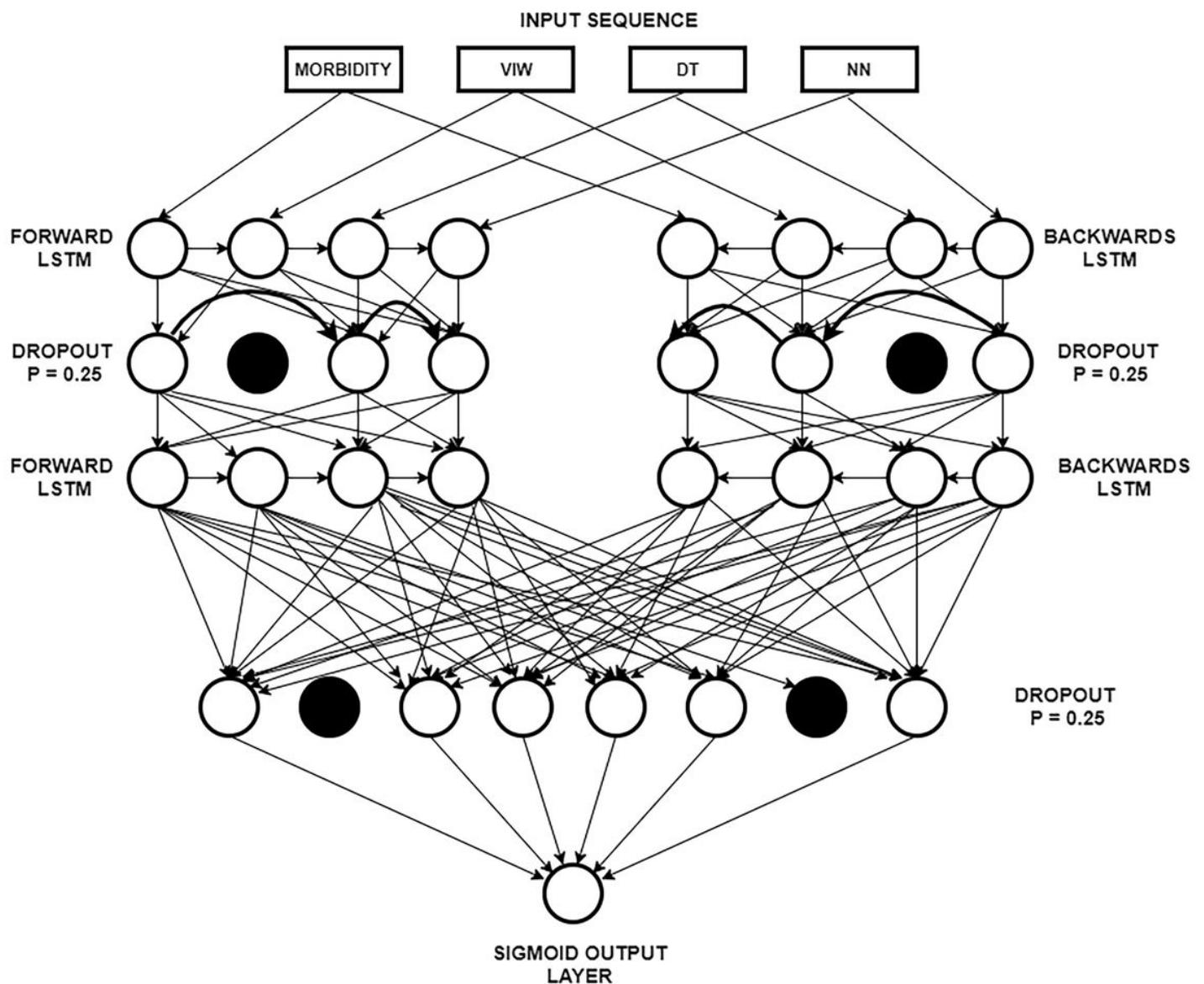


Figure 5

Model Architecture for Stack of 2 LSTMs classification Model



```
[SNOMEDCT[5217008] # Stiff-man syndrome (disorder)
, SNOMEDCT[9152004] # Nomeus gronovii (organism)
, SNOMEDCT[9247000] # Xylaria polymorpha (organism)
, SNOMEDCT[11446009] # Strobilomyces floccopus (organism)
, SNOMEDCT[12106005] # Lighthouse man (occupation)
, SNOMEDCT[13723001] # Blood group antigen Man (substance)
, SNOMEDCT[18184003] # Floatation man (occupation)
, SNOMEDCT[21198001] # Carbonation man (sugar refining) (occupation)
, SNOMEDCT[51658002] # Make-up man (stage/studio) (occupation)
, SNOMEDCT[59808009] # Blood group antibody Man (substance)
, SNOMEDCT[61545008] # Fermenting-room man (occupation)
, SNOMEDCT[68679005] # Stonework layout man (occupation)
, SNOMEDCT[77389004] # Anagallis arvensis (organism)
, SNOMEDCT[83903000] # Man lift, device (physical object)
, SNOMEDCT[10057001] # ...
```

Figure 7

Truncated output from SNOMED-CT search for "Man"

```
[SNOMEDCT[34041001] # Achard-Thiers syndrome (disorder)
, SNOMEDCT[50110003] # Cervical smear - endocervical cells absent (finding)
, SNOMEDCT[52832001] # Testicular feminization (disorder)
, SNOMEDCT[74715005] # Battered woman syndrome (finding)
, SNOMEDCT[95925006] # Psychologically abused woman (finding)
, SNOMEDCT[103610009] # Abnormal presence of endometrial cells in specimen from a menopausal woman (finding)
, SNOMEDCT[105438007] # Elderly woman (person)
, SNOMEDCT[125155008] # Endometrial cells, cytologically benign, in a postmenopausal woman (finding)
, SNOMEDCT[183863001] # Referral to well woman clinic (procedure)
, SNOMEDCT[185802005] # Attends well woman monitoring (finding)
, SNOMEDCT[185804006] # Well woman monitoring default (finding)
, SNOMEDCT[185805007] # Well woman monitoring first letter (procedure)
, SNOMEDCT[185806008] # Well woman monitoring second letter (procedure)
, SNOMEDCT[185807004] # Well woman monitoring third letter (procedure)
, SNOMEDCT[185808009] # Well woman monitoring verbal invite (procedure)
, SNOMEDCT[185809001] # Well woman monitoring telephone invite (procedure)
, SNOMEDCT[185810006] # Well woman monitoring deleted (finding)
, SNOMEDCT[185811005] # Well woman monitoring check done (finding)
```

Figure 8

Truncated output from SNOMED-CT search for "Woman"