autoNLP: NLP Feature Recommendations for Text Analytics Applications

Abstract

While designing machine learning based text analytics applications, often, NLP data scientists manually determine which NLP features to use based upon their knowledge and experience with related problems. This results in increased efforts during feature engineering process and renders automated reuse of features across semantically related applications inherently difficult. In this paper, we argue for standardization in feature specification by outlining structure of a language for specifying NLP features and present an approach for their reuse across applications to increase likelihood of identifying optimal features.

Introduction

For an ever increasing spectrum of applications (e.g., medical text analysis, opinion mining, sentiment analysis, social media text analysis, customer intelligence, fraud analytics etc.) mining and analysis of unstructured natural language text data is necessary BIBREF0, BIBREF1, BIBREF2.

One of key challenge while designing such text analytics (TA) applications is to identify right set of features. For example, for text classification problem, different sets of features have been considered in different works (spanning a history of more than twenty years) including `bag of words', `bag of phrases', `bag of n-grams', `WordNet based word generalizations', and `word embeddings' BIBREF3, BIBREF4, BIBREF5, BIBREF6, BIBREF7. Even for recent end-to-end designs using deep neural networks, specification of core features remains manually driven BIBREF8, BIBREF9. During feature engineering, often data scientists manually determine which features to use based upon their experience and expertise with respect to the underlying application domain as well as state-of-the-art tools and techniques.

Different tools (e.g., NLTK BIBREF10, Mallet BIBREF11, Stanford CoreNLP BIBREF12, Apache OpenNLP BIBREF13, Apache Lucene BIBREF14, etc.) available to a NLP data scientist for TA application design and development often differ in terms of support towards extraction of features, level of granularity at which feature extraction process is to be specified; and these tools often use different programing vocabularies to specify semantically equivalent features.

Currently, there is no generic method or approach, which can be applied during TA application's design process to define and extract features for any arbitrary application in an automated or semi-automated manner. Even there is no single way to express wide range of NLP features, resulting into increased efforts during feature engineering which has to start new for each data scientist and automated reuse of features across semantically similar or related applications designed by different data scientists is difficult. This also hinders foundational studies on NLP feature engineering including why certain features are more critical than others.

In this paper, we aim to present an approach towards automating NLP feature engineering. We start with an outline of a language for expressing NLP features abstracting over the feature extraction process, which often implicitly captures intent of the NLP data scientist to extract specific features from given input text. We next discuss a method to enable automated reuse of features across semantically related applications when a corpus of feature specifications for related applications is available. Proposed language and system would help achieving reduction in manual effort towards design and extraction of features, would ensure standardization in feature specification process, and could enable effective reuse of features across similar and/or related applications.

Life Cycle View

Figure FIGREF1 depicts typical design life cycle of a (traditional) ML based solution for the TA applications, which involves steps to manually define relevant features and implement code components to extract those feature from input text corpus during training, validation, testing and actual usage of the application. In traditional ML based solutions, feature interactions also need to be explicitly specified, though this step is largely automated when using deep neural network based solutions BIBREF9.

As the process of defining features is manual, prior experience and expertize of the designer affects which features to extract and how to extract these features from input text. Current practice lacks standardization and automation in feature definition process, provides partial automation in extraction process, and does not enable automated reuse of features across related application.

Next, let us consider scenarios when features are specified as elements of a language. Let us refer to this language as NLP Feature Specification Language (nlpFSpL) such that a program in nlpFSpL would specify which features should be used by the underlying ML based solution to achieve goals of the target application. Given a corpus of unstructured natural language text data and a specifications in the nlpFSpL, an interpreter can be implemented as feature extraction system (FExSys) to automatically generate feature matrix which can be directly used by underlying ML technique.

In contrast to the life-cycle view in the Figure FIGREF1, this would result into refined solution life cycle of ML based TA applications as depicted in the Figure FIGREF2.

NLP Feature Specification Language ::: Meta Elements

Figure FIGREF4 specifies the meta elements of the nlpFSpL which are used by the FExSys while interpreting other features.

Analysis Unit (AU) specifies level at which features have to be extracted. At Corpus level, features are extracted for all the text documents together. At Document level, features are extracted for each document in corpus separately. At Para (paragraph) level Features are extracted for multiple sentences constituting paragraphs together. At Sentence level features to be extracted for each sentence. Figure FIGREF6 depicts classes of features considered in nlpFSpL and their association with different AUs.

Syntactic Unit (SU) specifies unit of linguistic features. It could be a `Word' or a `Phrase', or a `N-gram' or a sequence of words matching specific lexico-syntactic pattern captured as `POS tag pattern' (e.g., Hearst pattern BIBREF15) or a sequence of words matching specific regular expression `Regex' or a combination of these. Option Regex is used for special types of terms, e.g., Dates, Numbers, etc. LOGICAL is a Boolean logical operator including AND, OR and NOT (in conjunction with other operator). For example, Phrase AND POS Regex would specify inclusion of a `Phrase' as SU when its constituents also satisfy 'regex' of `POS tags'. Similarly, POS Regex OR NOT(Regex) specifies inclusion of sequence of words as SU if it satisfies `POS tag Pattern' but does not match pattern specified by character `Regex'. Note that SU can be a feature in itself for document and corpus level analysis.

Normalize Morphosyntactic Variants: If YES, variants of words including stems, lemmas, and fuzzy matches will be identified before analyzing input text for feature exaction and would be treated equivalent.

NLP Feature Specification Language ::: Feature Types ::: Linguistic Features

Figure FIGREF8 depicts two levels of taxonomy for features considered as linguistic.

To illustrate, let us consider context based features: Table FIGREF9 gives various options which need to be specified for directing how context for an SU should be extracted. For example, Context_Window := [2, Sentence] will extract all tokens within current sentence, which are present within a distance of 2 on both sides from the current SU. However, Context_Window := [2, Sentence]; POSContext := NN$\mid $VB will extract only those tokens within current sentence, which are present within a distance of 2 on both sides from the current SU and have POS tag either NN (noun singular) or VB (verb, base form).

Table FIGREF10 illustrates how to specify the way head directionality of current SU should be extracted.

NLP Feature Specification Language ::: Feature Types ::: Semantic Similarity and Relatedness based Features

Semantic similarity can be estimated between words, between phrases, between sentences, and between documents in a corpus. Estimation could either be based upon corpus text alone by applying approaches like vector space modeling BIBREF16, latent semantic analysis BIBREF17, topic modeling BIBREF18, or neural embeddings (e.g., Word2Vec BIBREF19 or Glove BIBREF20) and their extensions to phrase, sentence, and document levels. Otherwise it can be estimated based upon ontological relationships (e.g., WordNet based BIBREF21) among concept terms appearing in the corpus.

NLP Feature Specification Language ::: Feature Types ::: Statistical Features

Figure FIGREF13 depicts different types of statistical features which can be extracted for individual documents or corpus of documents together with methods to extract these features at different levels.

In particular, examples of distributions which can be estimated include frequency distributions for terms, term distributions in topics and topic distributions within documents, and distribution of term inter-arrival delay, where inter arrival delay for a term measures number of terms occurring between two successive occurrences of a term.

Illustration of nlpFSpL Specification

Let us consider a problem of identifying medical procedures being referenced in a medical report.

Sample Input (Discharge summary): ““This XYZ non-interventional study report from a medical professional. Spontaneous report from a physician on 01-Oct-1900. Patient condition is worsening day by day. Unknown if it was started before or after levetiracetam initiation. The patient had a history of narcolepsy as well as cataplexy. Patient condition has not recovered. On an unknown date patient was diagnosed to have epilepsy. The patient received the first dose of levetiracetam for seizure."

Table TABREF14 shows specification of features in nlpFSpL.

Table TABREF15 and Table TABREF16 contain corresponding feature matrix (limited to first sentence and does not involve any preprocessing of input text).

NLP Feature Reuse across TA Applications

Next let us consider a case for enabling automated reuse of feature specifications in nlpFSpL across different semantically related applications.

NLP Feature Reuse across TA Applications ::: Illustrative Example

To illustrate that semantically different yet related applications may have significant potential for reuse of features, let us consider the problem of event extraction, which involves identifying occurrences of specific type of events or activities from raw text.

Towards that, we analysed published works on three different types of events in different domains as described next:

Objective of this study is to design a ML model for identifying if there exist mentions of one of the nine types of bio-molecular interactions in (publicly available) Biomedical data. To train SVM based classifier, authors use GENETAG database, which is a tagged corpus for gene/protein named entity recognition. BioNLP 2009 shared task test-set was used to estimate performance of the system. Further details can be found at BIBREF22.

Objective of the study was to design ML model for enabling automated detection of specific financial events in the news text. Ten different types of financial events were considered including announcements regarding CEOs, presidents, products, competitors, partners, subsidiaries, share values, revenues, profits, and losses. To train and test SVM and CRF based ML models, authors used data set consisting of 200 news messages extracted from the Yahoo! Business and Technology newsfeeds, having financial events and relations manually annotated by 3 domain experts. Further details can be found at BIBREF23.

Objective of the study was to design an ML based system for extracting open domain calendar of significant events from Twitter-data. 38 different types of events were considered for designing the system. To train the ML model, an annotated corpus of 1000 tweets (containing 19,484 tokens) was used and trained model was tested on 100 million most recent tweets. Further details can be found at BIBREF24.

Table TABREF21 below depicts classes of features selected by authors of these works (as described in the corresponding references above) to highlight the point that despite domain differences, these applications share similar sets of features. Since authors of these works did not cite each other, it is possible that that these features might have been identified independently. This, in turn, supports the hypothesis that if adequate details of any one or two of these applications are fed to a system described in this work, which is designed to estimate semantic similarities across applications, system can automatically suggest potential features for consideration for the remaining applications to start with without requiring manual knowledge of the semantically related applications.

NLP Feature Reuse across TA Applications ::: Reuse Process

Figure FIGREF23 depicts overall process flow for enabling automated feature recommendations.

For a new text analytics application requiring feature engineering, it starts with estimating its semantic proximity (from the perspective of a NLP data scientist) with existing applications with known features. Based upon these proximity estimates as well as expected relevance of features for existing applications, system would recommend features for the new application in a ranked order. Furthermore, if user's selections are not aligned with system's recommendations, system gradually adapts its recommendation so that eventually it can achieve alignment with user preferences.

Towards that let us start with characterizing text analytics applications. A TA application's details should include following fields:

Text based description of an TA application (or problem). For example, “identify medical procedures being referenced in a discharge summary” or “what are the input and output entities mentioned in a software requirements specification”.

Analysis unit at which features are to be specified and training annotations are available, and ML model is designed to give outcomes. Options include word, phrase, sentence, paragraph, or document.

Specifies technical classification of the underlying ML challenge with respect to a well-defined ontology. E.g., Classification (with details), Clustering, etc.

Specifies how performance of the ML model is to be measured - again should be specified as per some well defined ontology.

Knowledge base of text analytics applications contains details for text analytics applications in above specified format. Each application is further assumed to be associated with a set of Features (or feature types) specified in nlpFSpL together with their relevance scores against a performance metric. Relevance score of a feature is a measure of the extent to which this feature contributes to achieve overall performance of ML model while solving the underlying application. Relevance score may be estimated using any of the known feature selection metrics BIBREF25.

To specify knowledge base formally, let us assume that there are m different applications and k unique feature specifications across these applications applying same performance metric. Let us denote these as follows: $APPS=\lbrace App_1,\ldots , App_m\rbrace $ and ${\mathit {\Theta }}_F$ = $\left\lbrace F_1,F_2,\dots ,F_k\right\rbrace $ respectively. Knowledge base is then represented as a feature co-occurrence matrix $PF_{m\times k}$ such that $PF[i,j] = \delta _{i,F_j}$ is the relevance score of j^{th} feature specification ($F_j\in \mathit {\Theta }_F$) for i^{th} application $App_i\in APPS$.

NLP Feature Reuse across TA Applications ::: Measuring Proximity between Applications

To begin, for each text based field in each TA application, pre-process text and perform term normalization (i.e., replacing all equivalent terms with one representative term in the whole corpus) including stemming, short-form and long-form (e.g., ‘IP’ and ‘Intellectual Property’), language thesaurus based synonyms (e.g., WordNet based `goal' and `objective').

NLP Feature Reuse across TA Applications ::: Measuring Proximity between Applications ::: Identify key Terms

Thereafter, we identify potential `entity-terms' as `noun-phrases' and `action terms' as `verb-phrases' by applying POS-Tagging and Chunking. E.g., In sentence – “This XYZ non-interventional study report is prepared by a medical professional”, identifiable entity terms are “this XYZ non-interventional study report” and “medical professional” and identifiable functionality is `prepare'.

NLP Feature Reuse across TA Applications ::: Measuring Proximity between Applications ::: Generate Distributed Representations

Analyze the corpus of all unique words generated from the text based details across all applications in the knowledge base. Generally corpus of such textual details would be relatively small, therefore, one can potentially apply pre-trained word embeddings (e.g., word2vec BIBREF19 or Glove BIBREF20). Let $v(w)$ be the neural embedding of word w in the corpus. We need to follow additional steps to generate term-level embeddings (alternate solutions also exist BIBREF26): Represent corpus into Salton's vector space model BIBREF16 and estimate information theoretic weighing for each word using BM25 BIBREF27 scheme: Let $BM25(w)$ be the weight for word w. Next update word embedding as $v(w)\leftarrow BM25(w)\times v(w)$. For each multi-word term $z=w_1\dots w_n$, generate term embedding by averaging embeddings of constituent words: $v(z)\leftarrow \Sigma _{i=1}^{i=n}v(w_i)$.

In terms of these embeddings of terms, for each text based field of each application in the knowledge base, generate field level embedding as a triplet as follows: Let f be a field of an application in $APPS$. Let the lists of entity-terms and action-terms in f be $en(f)$ and $act(f)$ respectively. Let remaining words in f be: $r(f)$. Estimate embedding for f as: $v(f)$=$[v(en(f))$, $v(act(f))$, $v(r(f))]$, where $v(en(f))$=$\Sigma _{z\in en(f)} v(z)$, $v(act(f))$=$\Sigma _{z\in act(f)}v(z)$, and $v(r(f))$=$\Sigma _{z\in r(f)}v(z)$.

NLP Feature Reuse across TA Applications ::: Measuring Proximity between Applications ::: Estimating

Proximity between Applications

After representing different fields of an application into embedding space (except AU), estimate field level similarity between two applications as follows: Let $[X_i^{en}$, X^{act}_i, $X_i^{r}]$ and $[X_j^{en}$, X^{act}_j, $X_j^{r}]$ be the representations for field f for two applications App_i, App_j $\in APPS$. In terms of these, field level similarity is estimated as $\Delta _{f}(App_i,App_j)$ = $[\Delta _{en}({f_{i}, f_j})$, $\Delta _{act}({f_{i}, f_j})$, $\Delta _{r}({f_{i}, f_j})]$, where $\Delta _{en}({f_{i}, f_j})$ = 0 if field level details of either of the applications is unavailable else $\Delta _{en}({f_{i}, f_j})$ = $cosine(X_i^{en}$, $X_j^{en})$; etc.

For the field - AU, estimate $\Delta (au_i,au_j)$ = $1 \textit { if analysis units for both applications are same i.e., } au_i = au_j$ else 0.

In terms of these, let $\Delta (App_i,App_j)$ = $[\Delta _{en}({bd_{i}, bd_j})$, $\Delta _{act}({bd_{i}, bd_j})$, $\Delta _{r}({bd_{i}, bd_j}),$ $\Delta _{en}({dd_{i}, f_j})$, $\ldots $, $\Delta ({au_{i}, au_j})]$ be overall similarity vector across fields, where bd refers to the field `problem description' etc. Finally, estimate mean similarity across constituent fields as a proximity between corresponding applications.

NLP Feature Reuse across TA Applications ::: Feature Recommendations

Let $NewP$ be new application for which features need to be specified in nlpFSpL. Represent fields of $NewP$ similar to existing applications in the knowledge base (as described earlier in the Section SECREF30).

Next, create a degree-1 ego-similarity network for $NewP$ to represent how close is $NewP$ with existing applications in $APPS$. Let this be represented as a diagonal matrix $\Delta _{m\times m}$ such that $\Delta [r,r] = \alpha _i =$ proximity between $NewP$ and i^{th} application in the knowledge base (by applying steps in the Section SECREF31).

Thereafter, let $NorSim_{m\times k}=\Delta _{m\times m} \times PF_{m \times k}$ such that $NorSim[i,j]= \alpha _i \delta _{i,F_j}$ measures probable relevance of feature F_j for $NewP$ w.r.t. performance metric M based upon its relevance for $App_i \in APPS$. When there are multiple applications in $APPS$, we need to define a policy to determine collective probable relevance of a feature specification in $\mathit {\Theta }_F$ for $NewP$ based upon its probable relevance scores with respect to different applications.

To achieve that, let $Relevance$ $(NewP, f_j)$ be the relevance of f_j for $NewP$ based upon a policy, which can be estimated in different ways including following:

Next, consider there different example policies:

Weakest relevance across applications is considered: $Relevance(NewP,F_j) = \min _{i\in 1..m}{NorSim[i,j]}$

Strongest relevance across applications is considered: $Relevance(NewP,F_j) = \max _{i\in 1..m}{NorSim[i,j]}$

Most likely relevance across applications is considered: $Relevance(NewP,F_j) = \frac{1}{m}\Sigma _{i\in 1..m}{NorSim[i,j]}$

Rank feature specifications in $\mathit {\Theta }_F$ decreasing order based upon $Relevance(NewP,.)$, which are suggested to the NLP Data Scientist together with the supporting evidence.

NLP Feature Reuse across TA Applications ::: Continuous Learning from User Interactions

There are two different modes in which user may provide feedback to the system with respect to recommended features: one where it ranks features differently and second where user provides different relevance scores (e.g., based upon alternate design or by applying feature selection techniques). Aim is to use these feed-backs to learn an updated similarity scoring function $\Delta _{new}:APPS \times APPS$ $\rightarrow $ $[0,1]$.

In relation to $NewP$, let $Rank:{\mathrm {\Theta }}_F\times \left\lbrace system,user\right\rbrace \rightarrow \lbrace 1,\dots ,k\rbrace ~$ return rank of a feature and $Rel:{\mathrm {\Theta }}_F\times \left\lbrace system,user\right\rbrace \rightarrow [0,1]$ return relevance score of a feature based upon type – `system' or `user'.

Next, for each $App\in APPS$, let ${Ch}\left[App\right]$ $\leftarrow $ $\emptyset $ be a hash table with keys as application ids and values as list of numbers estimated next. Also let $NewSim_{FE}\left[.\right]\leftarrow 0$ contain updated similarity scores between $NewP$ and existing applications in $APPS$.

For each feature specification $f\in \Theta _F$, determine whether `user' given rank is different from `system' given rank, i.e., $Rank(f,`system^{\prime }) \ne Rank(f,`user^{\prime })$. If so, execute steps next.

Let $\mathit {Bind}(f_j) \subseteq APPS$ be the list of applications, which contributed in estimating collective relevance for feature $f_j \in \Theta _F$. For example, when aggressive or conservative policy is considered, $\mathit {Bind}(f_j)$ = $\lbrace App_r \mid Relevance(NewP,f_j)$ = $\mathit {NorSim[r,j]}\rbrace $.

For each $App_i\in Bind\left(f_j\right)$: Add x to $Ch[App_i]$, where x is estimated as follows: If user provides explicit relevance scores for App_i,

Otherwise if user re-ranks features

For each $App_i\in Bind\left(f_j\right)$: $NewSim_{FE}\left[App_i\right]\leftarrow Average(Ch[App_i])$. If $|NewSim_{FE}[App_i]$-${\alpha }_i|$ $\ge \epsilon {\alpha }_i$ i.e., when the difference between old and new similarity scores is more than $\epsilon $ fraction of original similarity, add $(\Delta (NewP,App_i),NewSim_{FE}[App_i])$ to training set Tr_{rpls} so that it used to train a regression model for $\Delta _{new}(.,.)$ by applying partial recursive PLS BIBREF28 with $\Delta (NewP,App_i)$ as set of predictor or independent variables and $NewSim_{FE}[App_i]$ as response variable. Existing proximity scores between applications in $APPS$ (ref. Section SECREF31) are also added to training set Tr_{rpls} before generating the regression model.

Note that $\epsilon $ is a small fraction $>$ 0 which controls when should similarity model be retrained. For example, $\epsilon = 0.05$ would imply that if change in similarity is more than 5% only then it underlying model should use this feedback for retraining.

Conclusion

In this paper, we have presented high level overview of a feature specification language for ML based TA applications and an approach to enable reuse of feature specifications across semantically related applications. Currently, there is no generic method or approach, which can be applied during TA applications' design process to define and extract features for any arbitrary application in an automated or semi-automated manner primarily because there is no standard way to specify wide range of features which can be extracted and used. We considered different classes of features including linguistic, semantic, and statistical for various levels of analysis including words, phrases, sentences, paragraphs, documents, and corpus. As a next step, we presented an approach for building a recommendation system for enabling automated reuse of features for new application scenarios which improves its underlying similarity model based upon user feedback.

To take this work forward, it is essential to have it integrated to a ML platform, which is being used by large user base for building TA applications so that to be able to populate a repository of statistically significant number of TA applications with details as specified in Section SECREF5 and thereafter refine the proposed approach so that eventually it rightly enables reuse of features across related applications.