Improving Information Retrieval Systems using the Stanford Dependencies representation

Chang Liu

Wayne State University

Fq8596@wayne.edu

Abstract—**in some simple searching scenarios, people prefer to retrieval information about a person or a product just by their names. Thus, the noun terms in the documents may play a more important role than other kind of terms, especially when they are acting a subject in the sentences. In this paper, we experimentally evaluated the effect of the Part-of-Speech (POS) tagging and the Stanford dependencies (SD) representation on Information Retrieval (IR) performance via three classic models (TF-IDF, Okapi BM25 and KL-divergence language model). We pre-processed the testing collection (TREC (AP 88-89)) for running POS and SD parsing tools, and reformatted the collection to get ready for the possible further modifications and researches on POS and SD in IR. Specifically, for POS， this paper amplified the term weights of noun terms in each models. For SD representation, we amplified the term weights of the terms which act subject in a sentence. Results show that doubling the term weights of subject terms can improve precision on P10 for the collection and a group of testing queries comparing to original BM25 and KL-divergence models. However, the schemes that three times or ten times amplifying those terms' weights negatively impacted the performance. And the weight scheme bias to noun term is not as good as the original models on precision or recall.**

***Keywords*— Information Retrieval, Part-of-Speech, Stanford dependencies**

I. INTRODUCTION

Many higher-level Natural Language Processing (NLP) techniques are widely used in linguistic applications. However, these methods increase the processing and storage cost dramatically. This makes them hard to be used in Information Retrieval systems on large and dynamically updating collections. But in some scenarios, like a company's archives or conversation records, the collections aren't huge and don't update frequently. It's affordable to periodically pre-process the collection using NLP to enhance the efficiency for IR. In addition, most of the searching queries in the aforementioned scenarios could be a costumer's name or a product's name, which both are noun words, and seem to be more relevant when they act as subjects in the sentences. In this paper, we propose two hypotheses: 1) increasing the term weights of noun terms can improve the performances of classical IR models. And 2) increasing the term weights of subject terms can improve the performances of classical IR models.

There are several challenges. The basic one is finding appropriate tools to recognize noun terms and subject terms. The Second one is how to coordinate these tools with IR toolkit. The hardest one is to pick the right collection and query set to show the benefit of proposed term weighting schemes.

We chose Stanford Log-linear Part-Of-Speech Tagger to identify the noun terms. A POS Tagger is a piece of software that reads text in some language and assigns parts of speech to each word, such as noun, verb, adjective, etc., This software is a Java implementation of the log-linear part-of-speech taggers. Here is an example sentence:

This is a sample sentence.

The POS tagger will give such outputs:

This/DT is/VBZ a/DT sample/NN sentence/NN

There are four types of noun taggers shown as following:

- NN Noun, singular or mass
- NNS Noun, plural
- NNP Proper noun, singular
- NNPS Proper noun, plural

The details of pre-processing the collection will be expatiated in part III.

We chose Stanford Dependencies (SD) parser to identify the subject terms. Dependency grammar describes linguistic structures directly in terms of dependencies between words themselves. Here is an example sentence:

Bell, based in Los Angeles, makes and distributes electronic, computer and building products.

For this sentence, the SD representation is:

nsubj(makes-8, Bell-1)
nsubj(distributes-10, Bell-1)
vmod(Bell-1, based-3)
nn(Angeles-6, Los-5)
prep_in(based-3, Angeles-6)
root(ROOT-0, makes-8)
conj_and(makes-8, distributes-10)
amod(products-16, electronic-11)
conj_and(electronic-11, computer-13)
amod(products-16, computer-13)
conj_and(electronic-11, building-15)
amod(products-16, building-15)
dobj(makes-8, products-16)
dobj(distributes-10, products-16

The relationship we used are "nsubj" and "nsubjpass". "nsubj" represents nominal subject. A nominal subject is a noun phrase which is the syntactic subject of a clause. The governor of this relation might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb, which can be an adjective or noun. "nsubjpass"

represents passive nominal subject. A passive nominal subject is a noun phrase which is the syntactic subject of a passive clause.

We chose Standard TREC AP 88-89 collection, and process it into the format that is easy to be extracted and run with standard parser. We had run the parsing process for more than two days using optimal parallel computing methods (6 machines×4 threads = 24 threads). Then programmed to filter out the noun terms and subject terms for each documents. And concatenated them with the original documents respectively. At the end of the pre-processing stage, we got five new collections including original collection with doubled noun terms, with doubled subject terms, with trebled subject terms, with ten times subject terms, and with both doubled noun terms and subject terms.

We separately configured three classic IR models (TF-IDF, Okapi BM25 and KL-divergence language model) for each of these collections. Then evaluated them with the same group of queries sample.

The results shown that doubling the term weights of subject terms can improve precision on P10 for the collection and a group of testing queries comparing to original BM25 and KL-divergence models. However, the schemes that three times or ten times amplifying those terms' weights negatively impacted the performance. And the weight scheme bias to noun term is not as good as the original models on precision or recall.

II. Related work

It has been shown that NLP techniques can be used effectively for IR tasks. Lexically, researchers use Part of Speech (POS) tagging to reduce the number of words indexed by the system [1]. However, When classic weighting scheme (like tf-idf) is factored over three POS tagging cases that are namely "No POS tagging", "POS tag with no history (i.e. 1-gram)", and "POS tag with one step history (i.e. 2-gram)", the experimental results show that the effect of POS tagging on information retrieval performance for Turkish is not significant better than original weighting scheme [7]. Pederson's work favoured noun phrases and adjective phrases to combine term weights. The results showed that adding these phrases and weighting schemes improved precision/recall over indexing terms alone [Pederson et. al. 97].

Syntactically, it has been difficult to develop retrieval models incorporating term dependence that show consistent improvements over those that assume term independence. Because a simpler retrieval model may be superior to a more sophisticated model depending on the query. To address this issue, Jae-Hyun Park, W. Bruce Croft and David A. Smith present a new term dependency model approach based on syntactic dependency parsing for both queries and documents [3]. They model four different types of relation-ships between syntactically dependent term pairs to perform inexact matching between documents and queries. And used a machine learning approach to find an optimal parameter setting for a combination of retrieval models. Their results on TREC collections show that the quasi-synchronous dependence model can improve retrieval performance and outperform a strong state-of-art sequential dependence baseline when use predicted optimal parameters. This project will draw lessons from their work, and only emphasize one kind of dependence relation, "nsubj", rather than considering the whole dependency Treebank of sentences.

Question answering system is another active field using dependency relations in the match module. For instance, subject-Verb Comparison. The system compares hypothesis subject and verb with retrieved sentence subject and verb that are identified through the nsubj and nsubjpass dependency relations. [2]

The Stanford typed dependencies representation was designed to provide a simple description of the grammatical relationships in a sentence that could easily be understood and effectively used by people without linguistic expertise who wanted to extract textual relations [4]. The open information extraction system TEXTRUNNER (Banko et al., 2007) also makes use of the SD graph representation: its first module uses the Stanford parser and the dependency tool to automatically identify and label trustworthy and untrustworthy extractions.

III. Project Description

A. Process

Step1: Learning to use Stanford Parser program package (vision 3.5.0).

a) Environment

The current version of the software requires Java 8 (JDK1.8) or later. It also requires a reasonable amount of memory (at least 100MB to run as a PCFG parser on sentences up to 40 words in length; typically around 500MB of memory to be able to parse similarly long typical-of-newswire sentences using the factored model).

b) Tagger model choice

For English, the bidirectional taggers are slightly more accurate, but tag much more slowly. We don't have tight speed requirement, and want high accuracy. Thus, we chose "bidirectional-distsim-wsj-0-18.tagger". It was trained on WSJ sections 0-18 using a bidirectional architecture and including word shape and distributional similarity features. It shows 97.28% correct on WSJ 19-21, and 90.46% correct on unknown words. Because the training collection is similar to our testing TREC collection. We have reason to believe that the bidirectional model will provide an accuracy higher than 90%.

c) SD parser model choice

We used the PCFG parser model, which is quick to load and run, and quite accurate. It takes a few seconds to load the parser data before parsing begins; continued parsing is quicker. To account for very long sentences in the documents, we used the flag "-mx600m" to give more memory to java.

Step2: Adjusting documents' format.

In TREC AP 88-89 collection, there are 686 txt files. And in each of these files includes about 100~300 documents in xml format. The content between labels <TEXT> and </TEXT> is the main body of the document. Some of documents start at the same line with label <TEXT>. Others start at the next line after

the label <TEXT>. We adjusted them to match the later format. So that we can extract all the documents' contents easily and correctly.

Step3: Running the parser on documents one by one.

We extracted one document from the uniformed collection files once a time. Restore a temporary copy for it. Run the POS tagger and SD parser on it. Then concatenated all the outputs with original document within the </TEXT> label, and made separating mark between original content and outputs.

We wrote a python program to execute above procedure for all original collection files. To boost the efficiency of this step, we employed 6 computers and run 4 threads in parallel. The whole procedure still cost more than 2 days.

Then we created a transitional collection with all outputs of POS tagger and SD parser. This transitional collection can be used to further researches, like extracting any sets of terms or phrases according to any arbitrary requirements related to POS or SD.

Step4: Filtering out the terms we need and concatenating them with original documents.

a) Create a new collection with doubled noun terms

Basing on the transitional collection, we extracted noun terms which have "NN", "NNS", "NNP" and "NNPS" taggers, and deleted other outputs and taggers.

b) Create three new collections with doubled, or trebled or ten times subject terms.

Basing on the transitional collection, we filtered out subject terms which occurred in the "nsubj" or "nsubjpass" pairs, and deleted other outputs and taggers. This will create a new collection with doubled subject terms. The trebled or tem times ones just need copy these terms certain times respectively.

c) Create a new collection with doubled noun terms and doubled subject terms.

We combined filtering results of noun terms and subject terms, and deleted other outputs and taggers.

We wrote another python program to execute above procedure with small modifications to work for different extracting requirements.

Step5: Indexing five collections separately.

We used version 4.12 of Lemur IR toolkit. And run the Lemur's BuildIndex program on each above collection.

Step6: Configuring three models for each of five collections.

To compare original models with proposed weighting scheme models, we chose RetEval program to construct the original models of TF-IDF, Okapi BM25 and KL-divergence with Dirichlet prior smoothing. Because in several home-works, their evaluation results are better.

For TF-IDF model, we only set the parameter of document term TF weighting method to be log-TF. So the original weighting function is as following:

$$W_{t,d} = \log(1 + tf_{t,d}) \times \log(N/df_t)$$

The doubled noun terms collection is equivalent to modify the model as:

$$W_{common-term,d} = \log(1 + tf_{t,d}) \times \log(N/df_t)$$
$$W_{noun-term,d} = \log(1 + 2 \times tf_{t,d}) \times \log(N/df_t)$$

The doubled subject terms collection is equivalent to modify the model as:

$$W_{common-term,d} = \log(1 + tf_{t,d}) \times \log(N/df_t)$$
$$W_{subject-term,d} = \log(1 + 2 \times tf_{t,d}) \times \log(N/df_t)$$

By analogy, we got models for trebled and ten times subject terms collections.

For Okapi BM25 model, we set the parameters as following: k_1=1; b=0.5; k_3=1. These are reasonable defaults used by Xapian which is an open source probabilistic information retrieval library. So the original weighting function is as following:

$$\mathrm{W} = \frac{(k_3+1)q}{(k_3+q)} \times \frac{(k_1+1)f}{\left(k_1(\mathrm{b}L + (1-b)) + f\right)} \times \log\frac{(N-n+0.5)}{(n+0.5)}$$

where

q is the within query frequency;

f is the within document frequency;

n is the number of documents in the collection indexed by this term;

N is the total number of documents in the collection;

L is the normalised document length.

The doubled subject terms collection is equivalent to modify the model as:

For common terms, didn't change anything.

For subject terms, its f was doubled.

For both common and subject terms, the L is increased to the sum of original L and the number of subject terms.

By analogy, we got models for other created collections.

For KL-divergence retrieval model, we only set the parameter of smooth method to be dir. So the original weighting function is as following:

$$\mathrm{W} = \mathrm{p}(w|\theta_Q) \log\left(1 + \frac{f}{\mu p(w|C)}\right) + \log\frac{\mu}{\mu + L}$$

We set $\mu = 1250$ according to the results of homework-3.

The doubled subject terms collection is equivalent to modify the model as:

For common terms, didn't change anything.

For subject terms, its f was doubled.

For both common and subject terms, the L is increased to the sum of original L and the number of subject terms.

By analogy, we got models for other created collections.

We created 18 configuration files manually to run above models for original collection and modified collections.

Step7: Run testing queries and evaluating

We used IR evaluation tool (trec_eval) to measure the P10, P20, and MAP values of aforementioned models for 18 times. The results will be explained in part IV.

B. Workload

In this project, the workload mainly reflected in understanding and using the parser program package and coordinating it with multiple IR models.

TABLE I

task	Workload		
	code	**configuration**	**Running cost**
Pre-process	52 lines	5 files	
Tagging /Parsing	280 lines		6 computers 24 threads > 2 days
Retrieval /Evaluating		24 files	

IV. Results and Analysis

In the perspective of IR models, KL-divergence retrieval model with Dirichlet prior smoothing method significantly outperforms than TF-IDF and BM25 models on all three criteria including P10, P20 and MAP. And Okapi BM25 model is better than TF-IDF model overall. This result is as expected. Because we considered more factors in KL-divergence retrieval model and added smoothing method in it.

In the perspective of proposed hypothesis, doubling noun terms' weights didn't improve the performances of original models. This result may be caused by queries choice. The most queries samples we used are "noun+verb" phrases. If we have a group of noun queries, the benefit of this weighting scheme may show.

On the other hand, **doubling subject terms' weights did show higher P10 value than the original BM25 and KL models as expected**. The reason why above improvement didn't show on TF-IDF model is that we did not only double the subject terms' weight, but also increased the document length. However, TF-IDF doesn't normalize document length like BM25 and KL models do.

Furthermore, because of above improvement, we also increase the subject terms' weights to 3 times and 10 times. But the results became worse than just doubling. Treating the coefficient of the subject term frequency as a parameter, we concluded that 2 is its optimal value here. And combining the weighting scheme of doubling noun terms and doubling subject terms, on the contrary, reduced the evaluation criteria.

In the perspective of evaluation criteria, the precision criteria are better than MAP for all models. Because there was a tendency to consider more precision than recall when we proposed these hypothesis. Various factors may impact on the evaluation results. The detail data will be shown in following figures.

A. *TF-IDF model*

"orig" represents original tf-idf model. "nn" represents the modified model by doubling the term weights of noun terms. "ns" represents the modified model by doubling the term weights of subject terms. The blue line is the wave of MAP value. The red line is the wave of P10 value, and the gray line is the wave of P20 value.

The results data are as shown in Fig. 1.

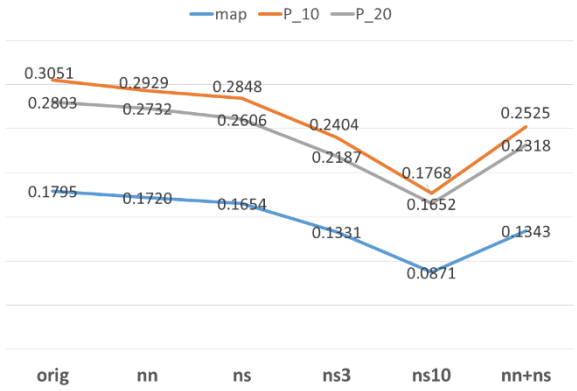

Fig. 1 Evaluation results of TF-IDF model with different collections

B. *Okapi BM25 model*

The P10 value of doubling subject terms' frequencies model is higher than original BM25 model as shown in red (0.3919 > 0.3899). While, their P20 and MAP are very nearly the same.

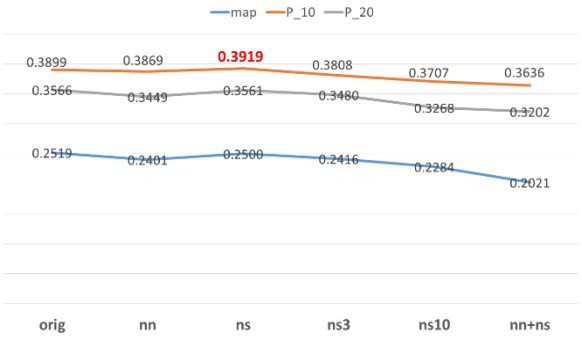

Fig. 2 Evaluation results of BM25 model with different collections

C. *KL-divergence retrieval model*

The P10 value of doubling subject terms' frequencies model is higher than original KL model as shown in red (0.4081 > 0.4071).

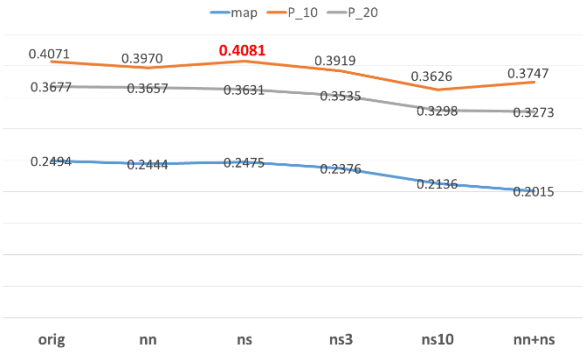

Fig. 3 Evaluation results of KL model with different collections

V. Conclusion

In this paper, the experimental testing results show that doubling the term weights of subject terms can improve precision on P10 comparing to original BM25 and KL-divergence models. But 3 times or 10 times subject terms' weights will reduce the performance.

The models which normalize the documents length and consider smoothing perform better. And it seems better to punish the increasing of document length when add certain terms' frequency

We need employ parallel computing technology to accelerate speed of NLP in IR

Having transitional collection, there are other dependency relationships that may be useful in IR for further researches. For instance, agent relation or adjectival modifier relation. An agent is the complement of a passive verb which is introduced by the preposition \by" and does the action, e.g. "The man has been killed by the police" agent(killed, police). An adjectival modifier of an NP is any adjectival phrase that serves to modify the meaning of the NP, e.g "Sam eats red meat" amod(meat, red) .

VI. References

[1] "Improving Information Retrieval Systems using Part of Speech Tagging", Abdur Chowdhury, M. Catherine McCabe, Digital Repository at the University of Maryland, 1998;

[2] "A Hybrid Question Answering System based on Information Retrieval and Answer Validation" Partha Pakray1, Pinaki Bhaskar1, Somnath Banerjee1, Bidhan Chandra Pal1, Sivaji Bandyopadhyay1, Alexander Gelbukh2, CLEF 2011 Workshop on QA4MRE, 2011;

[3] "A Quasi-Synchronous Dependence Model for Information Retrieval" Jae-Hyun Park, W. Bruce Croft and David A. Smith, CIKM '11 Proceedings of the 20th ACM international conference on Information and knowledge management, Pages 17-26

[4] "The Stanford typed dependencies representation" Marie-Catherine de Marneffe, Christopher D. Manning, Proceeding CrossParser '08 Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation Pages 1-8, 2008;

[5] "Stanford typed dependencies manual" Marie-Catherine de Marneffe and Christopher D. Manning, 2008;

[6] "Universal Stanford Dependencies: A cross-linguistic typology" Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, Christopher D. Manning, Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014);

[7] "The effect of Part-of-Speech tagging on IR performance for Turkish", B. Taner Dinçer, Bahar Karao lan, Computer and Information Sciences - ISCIS 2004.

[8] "Natural Language Processing in Information Retrieval", Thorsten Brants, Google Inc.