wordnet based self-transfer learning

転移学習とfew-shotの研究をしたいです。

具体的なテーマは wordnet based self-transfer learning.

1. Motivation

1.1 目標

転移学習で、データを利用する効率を上げることで、few-shotタスクを解決する。

1.2 考え

これは今読んでいる論文 graph network からの想いです。

知識には構造性(オーダと階層性)を持っています。わかりやすくため、私は今のモデルを四つの種類に分けています。

- **Type1**

  もっとも簡単なモデル、たとえばlogistics回帰あるいは浅いネットワークは、データ(入力)とモデル両方とも知識の構造性を考慮に入れていません。つまりモデルの帰納バイアスが弱いということです。

- **Type2**

  次に入力データは構造性を持っていますが、モデルには構造性を持っていない場合です。例えば、各単語にwordnetなどの知識ベースのラベルを付くことで、タスクに構造的な知識を導入する手法です。モデルの帰納バイアスが弱いです。

- **Type3**

  三つ目はモデルとデータ両方とも構造性を持つ、そしてモデルの帰納バイアスが中レベルの場合です。たとえば、LSTMを用いるタスクはデータのオーダを考えていますので、モデルとデータが弱い構造性を持っています。

- **Type4**

  四つめはモデルとデータ両方とも強い構造性を持つ, つまりモデルの帰納バイアスが強い手法です。例えば、graph LSTMs モデルはオーダだけではなく、文中の依存木構造知識と文間の会話構造も考慮に入れています。

求めると、以下の表が得られます。

-	データに構造性？	モデルに構造性？	帰納バイアス
Type1	False	False	弱い
Type2	True	False	弱い
Type3	True(弱い)	True	中
Type4	True(強い)	True	強い

> 注 : 上三つの変数が互いに独立ではありません。制約条件があります。
>
> - モデルが構造性を持つと、データも必ず構造性を持つ。
> - モデルが構造性を持つと、帰納バイアスは弱いより以上です。
> - モデルが構造性を持たない場合では、帰納バイアスは弱いです。

そして、まとめによって、一つ経験的な結論ができます。モデルが構造性を持つ場合では、ネットワークモデルをgraph networkに抽象化すると、graphは動的です。つまり、モデルの抽象化graph networkは、入力によって変わることができます。たとえば、入力文の長さによってLSTMの graph network 長さが変わります。

> ですが、構造性を持つモデルのgraph networkは必ず動的とは言えません。つまり、モデルが構造性を持つことは、モデルの抽象化グラフが動的であることの必要条件です。
>
> 十分ではないの例を挙げます。たとえば、CNNの場合では、画像のサイズが定めている場合は、抽象化したモデルのグラフは変わりませんが、近いところのデータが関連性あるという強い帰納バイアスを前提にしていますので、モデルが構造化を持っています。

1.3 やりたいこと

転移学習で、Type4 よりさらに強い帰納バイアスモデルを作ることで、貴重なデータを利用する効率を上げたいです。

Type4 は文法的な構造化知識を利用していますが、私は語彙的な構造化知識をモデルの融合して、もっと強い帰納バイアスを持つモデルを作りたいです。

2. Method

2.1 手法について

以下は考えついた大体な手法です。おおまかだけで詳しいところをまだ考えていません。

まとめて言うと、wordnet中の語彙的な知識でモデルを構造化したいです。wordnetは認知的な情報をもっています。今のwordnetを利用する研究はほとんど **type2** のクラスに属しています。つまり、非構造化なモデルで構造化的な知識を処理することです。この方法には、もっと本質的な欠点があると思いますが、今考え付いたのは、データを利用する効率が低いということです。

wordnet 利用する方法が、self-transfer learning です。

具体的な方法が以下のとおりです。

- コーパスなかの単語を全文 wordnet によって上に遡ります。たとえば、「犬」は「動物」のヒポニムなので、遡ることによって、コーパス中の「犬」を全部「動物」に変換します。このように変換したコーパスは元より, エントロピーが少なく、学習しやすいと想定しています。そうするの目標は、知識の制約を弱めることによって、構造化知識の高い層の知識を精確的に得ることです。
- そして、遡る回数によって、いくつの「バージョン」が得られる。制約がもっとも弱い「バージョン」のコーパスから言語モデルをトレーニングします。そして、得たモデルのパラメータを分析することによって、新しいモデルを設計します。具体的な分析方法とか、設計する方法はまだ分かりませんが、設計したモデルは一つ特徴を持つはず、それは入力の語彙構造によって、graph networkの形が変わることです。
- 最後で、変わった構造を利用して次の「バージョン」をトレーニングします。

2.2 例

言語モデルを例にします。

例えば、「　×　が漏れています。」という例の中の「×」を予測したいタスクがあります。

普通のモデルの場合は、すべての単語が予備語になっています。Perplexityが高いです。

この方法を利用することで、まずは高い層での処理で「x」の範囲を「情報、液体」などの範囲に制約することができます（高い層が精確なので）。これで、予備語になっている単語の量が少なくなっている、それに応じて、Perplexityが低くなるはずだとおもいます。

ここで一番重要なのは　**「高い層での処理で「x」の範囲を「情報、液体」などの範囲に制約すること」**　です, この処理の本質は**「高い層処理の精確性」**によって連続的な分布を離散化することです。これによってタスクのエントロピーが減少します。

2.3 理論的な根拠

このやり方の理論的な根拠は：

- **非構造化モデルの場合**

  n個のunitsを持つ非構造化モデルのgraph networkは以下のように抽象化します：

  各unitの状態値の種類をm個に設定します。ならば、このタスクのエントロピーは：

  $H_1 = -n * log(1/m)$

  そして、もしi 番目のunitの状態値は確率分布 P に従うことを知れば、このタスクのエントロピーは：$H_2 = -\sum_i^m P_i(n-1)log(1/m) + P_i logP_i$　となります。

  $\triangle\ H = H_1 - H_2$

- **構造化モデルの場合**

  n個のunitsを持つ構造化モデルのgraph networkは以下のように抽象化します：

  各unitの値を決まることは高い層から低い層までたどり着くことが必要です。そして、一番低い層(葉、つまり最終の状態値)の数がmです。ここは、層数を k にします ($2^k = m$) 。

  知識を何も知らない状態では、タスクのエントロピーはうえと同じ：

  $H'_0 = -log(1/m)$

  もし、j 層で、一つのnodeに決めることができれば：

  $H'_1 = -\sum_i^{m^{n-1}*2^{k-j}} \frac{1}{(m^{n-1}*2^{k-j})} * log(\frac{1}{m^{n-1}} * \frac{1}{2^{k-j}}) - \sum_i^{m^{n-1}*(2^k-2^{k-j})} 0 = -(n-1)log(1/m)$

  H'_1 が上の H_1 と対応しています。

  そして、もじ j 層が決めたうえで、$j+1$ 層の分布が得られば、H'_2 も得られます。

  $\triangle\ H' = H'_1 - H'_2$

- そして、まで証明できていませんが、直感では

  $\triangle\ H' = \triangle\ H$ だと思います。

  以上は**仮想の**この研究をする理論的な根拠です。