マルチタスク学習による一貫性のある求人記事分類及び職種フレーズと記事見出し文の生成

西埜 徹　三沢 翔太郎　狩野 竜示　谷口 友紀　三浦 康秀　大熊 智子
富士ゼロックス株式会社
{nishino.toru, misawa.shotaro, kano.ryuji, taniguchi.tomoki, yasuhide.miura, ohkuma.tomoko}@fujixerox.co.jp

1 はじめに

Web 上の記事の量が爆発的に増大する昨今では, 記事の分類や記事の属性を表すキーフレーズ・記事見出し文は読み手が記事を取捨選択する際に重要な情報である. 表 1 に示す求人情報サイトの記事では, 読み手は職種毎の分類カテゴリおよび職種を表すフレーズから求人記事を絞り込み, 記事見出しを読んで関心ある記事を選択している. しかし記事分類や適切な職種フレーズ・見出し生成は記事作成者にとって負担であり, 負担軽減のため分類・文生成技術の重要性は高まっている.

本研究の目的は, 求人情報サイトの記事本文を入力として与えた時, 記事見出し文の生成, 職種を表すフレーズの生成および職種カテゴリに応じた記事分類を出力することである. Rush ら [1] を始め, 深層学習による生成型要約を用いた見出し生成は多く行われているが, 分類や職種フレーズ生成のタスクと見出し生成のタスクとで各個に生成する必要がある. それゆえ, 職種分類・職種フレーズと見出し文の間で異なる職種名が含まれるなど, 3 タスクの出力結果が一貫性に欠ける問題点が存在する.

本研究の貢献は以下の 2 点である. (1) マルチタスク学習の適用による見出し生成タスクと職種フレーズ生成タスク及び職種分類の 3 タスクを同時に解くモデルを実現する. (2) 3 タスク間の学習スケジューリング手法を提案する. マルチタスク学習には少ない教師データでの学習を促進し, 3 タスク共通の因子 (記事中の職種情報) を共通エンコーダ層で捕捉できる利点がある. マルチタスク学習による要約の研究として, Isonuma ら [2] は抽出型要約と文書分類のマルチタスク学習により精度を改善し, Bansal ら [3] は, 質問文生成とのマルチタスク学習により生成型要約の改善を実現した. それに対し本研究では, 共通して職種情報へ着目が必要な 3 タスク間のマルチタスク学習により, 職種に関して一貫性のある記事分類・職種フレーズ及び見出し文生成が期待される. また, 学習スケジューリング機構の導入によって難易度差のあるマルチタスク学習を促進し, さらなる精度向上が期待される.

2 対象データの特徴

本研究では求人情報サイト Wantedly [*1] の記事を扱う. 求人記事の例を表 1 の例に示す. 1 つの求人記事に対して 1 つの見出し文と職種分類カテゴリ (全 18 種), 職種を表すキーフレーズ (自由記述) で構成されている. 対象の求人記事データの特徴は以下の 3 点である.

教師データ量の少なさ　要約タスクでよく用いられる CNN/DailyMail データセット [4] や Gigaword データセット [*2] に比べ, 表 2 の通り本タスクの記事件数は約 5 万件と少なく, 学習時の課題の 1 つである.

表 1　Wantedly 社求人記事の例

【職種分類カテゴリ】 モバイルエンジニア
【職種フレーズ】 Android™ [*3] エンジニア
【見出し文】 新サービス・Wantedly People の Android エンジニア募集！
【本文】Wantedly では 2016 年 11 月から新サービス「Wantedly People」をリリースし、2017 年 7 月に新機能もリリースしました！急成長中の Wantedly People を開発する People チームでは現在「Android エンジニア」を募集しています。(以下略)

表 2　実験用データセットの統計量

全データ数	52,914	職種分類カテゴリ数	18
平均記事単語数	336.5	語彙数	32,972

3 タスク間の難易度差　職種フレーズは職種分類をさらに細分化したものであり, 見出し文の多くは職種フレーズの情報も含んだ文章である. それゆえ職種分類・職種フレーズ・見出し文の順にタスクの難易度が上昇すると言える.

3 タスクで共通する情報が含まれる点　職種分類と職種フレーズの双方に職種に関する情報が含まれるとともに, 大半の見出し文も職種に関する情報を含んでいる. 職種は読み手が最も関心を持つ情報の 1 つであり, 職種分類カテゴリと見出し文に含まれる職種が食い違う場合, 希望する職種の求人を探す際の障害となりうる. それゆえ, 職種の一貫性を保つことは本タスクの重要な課題の 1 つと言える.

3 提案手法

3.1 マルチタスク学習の特徴と効能

マルチタスク学習とは, 関連する複数のタスク間で共通する隠れ状態ベクトルを学習させ, より汎化したモデルの学習を実現する手法である. 深層学習においては, 一部の層のパラメータを複数タスクで共有する手法が主流である. Ruder ら [5] はマルチタスク学習が効果がある理由として次の 3 点を挙げている.

潜在的データ拡張　複数タスクでの学習ではモデルを学習するデータ数が拡張されるため, タスク固有のノイズの過学習を抑制し汎化する効果がある.

盗み聞き　タスク間に難易度差がある時, 簡単なタスクの情報から複雑なタスクの学習を促進できる.

Attention Focusing　タスク間で共通する特徴に着目することで, ノイズではなくタスクに関係ある特徴を捉える効果がある.

[*1] Wantedly はウォンテッドリー株式会社の商標です.

[*2] https://catalog.ldc.upenn.edu/LDC2012T21

[*3] Android は Google LLC. の商標です.

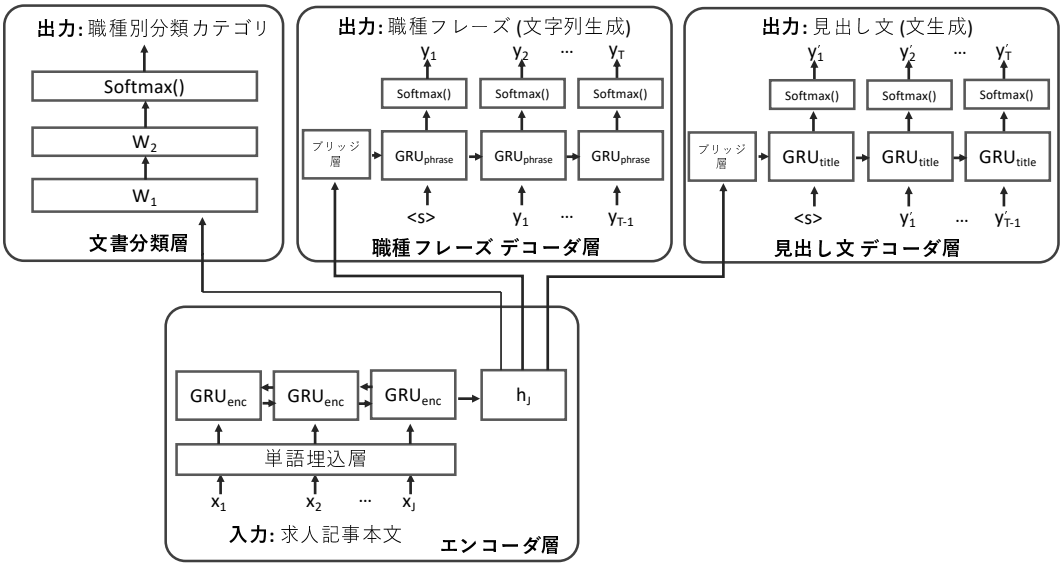

図 1　提案手法の概要

本研究で扱う求人記事データセットにおいては, 2 章で触れたように, (1) 教師データが 5 万件程度と少なく, 過学習が課題. (2) タスクに難易度差があるため学習促進が期待できる. (3) 職種分類カテゴリ・職種フレーズ・見出し文の全てにおいて “職種” が共通する特徴であり, 注意対象を共有することで的確かつ一貫して入力文書中の特徴を捉えられる. の 3 点の理由から, マルチタスク学習に適しているタスクと予想できる.

3.2　生成型要約

自然言語処理による要約手法は, 抽出型要約と生成型要約に大別できる. 入力文書の一部分を切り出して要約文を生成する抽出型要約の手法に対し, 生成型要約では入力文書の情報を元に新たな文を生成することで要約する. 記事見出しは本文の切り出しではなく本文を元に言い換え・省略を施した文章であるため, 抽出型要約より生成型要約が適したタスクと考えられる. そのため本研究では生成型要約による見出し生成を試みる.

本研究では, 近年の生成型要約研究の主流である Recurrent Neural Network (RNN) を用いた生成型要約手法をベースラインとして適用する.

エンコーダ・デコーダモデル　Rush らの RNN 生成型要約モデルは, エンコーダ及びデコーダの 2 つの RNN を組み合わせた構造をとる. エンコーダで入力記事を一度隠れ状態ベクトルに変換し, デコーダで隠れ状態ベクトルから出力文章へ変換することで文章から文章への系列変換が可能となる. ここで, 入力する記事本文を $x = \{x_1, x_2, ..., x_J\}$, 隠れ状態ベクトルを $h^e = \{h^e_1, h^e_2, ..., h^e_J\}$ 及び出力文章を $y = \{y_1, y_2, ..., y_T\}$ と表すと, エンコーダ・デコーダモデルは下式で表される.

$$h^e_t = \mathrm{RNN}_{enc}(x_t, h^e_{t-1}) \tag{1}$$

$$h^d_t = \mathrm{RNN}_{dec}(y_t, h^d_{t-1}, h^e_J) \tag{2}$$

$$P_{vocab,t} = \mathrm{softmax}(W_{d2v} h^d_t + b_{d2v}) \tag{3}$$

注意機構　また, Rush らは 注意機構を導入し, 長い系列の入力へ対応している. 注意機構はエンコーダの各段階の隠れ状態ベクトルを重み付き平均として計算し, 入力記事内の重要な情報を選別する. 注意機構を下式で表す.

$$e^e_{tj} = v^T \tanh(W_e h^e_j + W_d h^d_t + b_{attn}) \tag{4}$$

$$\alpha^e_{tj} = \mathrm{softmax}(e^e_{tj}) \tag{5}$$

$$c^e_t = \sum_j \alpha^e_{tj} h^e_j \tag{6}$$

$$\bar{h^d_t} = W_c([h^d_t, c^e_t] + b_c) \tag{7}$$

(5) 式では, デコーダの t 番目の出力を計算する際の隠れ状態ベクトルの重要度 α^e_{tj} を計算する. 次に入力記事の文脈ベクトル c^e_t を隠れ状態ベクトルの重み付き平均として算出する. (7) 式で文脈ベクトルとデコーダの t 番目の隠れ状態ベクトルと組み合わせ, 入力記事の重要部を重視した新たな隠れ状態ベクトル $\bar{h^d_t}$ を得る.

コピー機構　エンコーダ・デコーダモデルによる要約では, 教師データを元に作成した辞書の中からデコーダ層の出力する単語を決定する. それゆえ未出語に対応できない. この欠点に対応した手法に, コピー機構を採用した Pointer-Generator Model [6] が挙げられる.

$$p_{gen,t} = \sigma(W_c c^e_t + W_d h^d_t + W_y y_t + b_{ptr}) \tag{8}$$

$$P_t(w) = p_{gen,t} P_{vocab,t}(w) + (1 - p_{gen,t}) \sum_j \alpha^e_{tj} \tag{9}$$

コピー機構では, 入力として与える記事本文の単語列を辞書の拡張として扱う. デコーダで t 番目の単語を出力する際, (8) 式で求める $p_{gen,t}$ の確率で RNN の出力を元に単語を生成し, $1 - p_{gen,t}$ の確率で入力単語列をコピーした単語を出力する. コピー機構により辞書中に無い単語を含んだ要約文も生成が可能となる. また, Pointer-Generator Model では Coverage 機構と呼ばれる同一単語の反復を減らす手法が採用されている.

3.3　マルチタスク学習の構成

提案手法の概要を図 1 に示す. 提案手法は, 3 タスクで共通のエンコーダと, 各タスク固有の分類・デコーダ層からなる.

単語埋込層　単語を単語分散表現へと射影する. 単語埋込層はエンコーダへの入力とデコーダへの入力とで共有される.

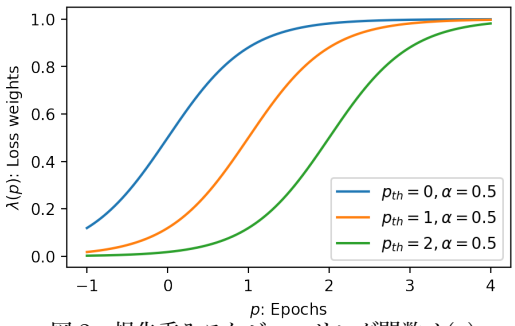

図 2　損失重みスケジューリング関数 $\lambda(p)$

エンコーダ層　注意機構付き 2 層双方向 GRU を使い入力文書を隠れ状態ベクトルに変換する．エンコーダ層は 3 タスク間で共有される．

文書分類層　多層パーセプトロンからなる．共有エンコーダ層で出力された隠れ状態ベクトルを入力し，文書の職種分類カテゴリを表す One-hot 表現を出力する．

デコーダ層　注意機構・コピー機構付き 2 層 GRU を使い，職種フレーズ及び見出し文を生成する．共有エンコーダ層の出力した隠れ状態ベクトルは，まず 1 層 NN からなるブリッジ層で各タスクごとに変換された後にデコーダ層に入力される．

マルチタスク学習の全体損失関数 L_{all} は，(10) 式のように各タスクの損失関数の重み付き和で定義される．L_{cla}, L_{phrase}, L_{title} はそれぞれ文書分類・職種フレーズ生成・見出し文生成の損失関数，λ_{cla}, λ_{phrase}, λ_{title} は 3.4 節で後述する各タスクの重み係数である．

$$L_{all} = \lambda_{cla} L_{cla} + \lambda_{phrase} L_{phrase} + \lambda_{title} L_{title} \quad (10)$$

3.4　マルチタスク学習の学習スケジューリング

記事の職種分類・職種フレーズ生成・見出し文生成の 3 タスク間には難易度の差が存在するため，盗み聞き効果により簡単なタスクの情報が複雑なタスクの学習の補助になると期待される．本研究ではマルチタスク学習の各タスクの学習スケジューリングを行い，盗み聞き効果を促進させる手法を提案する．

Kiperwasser ら [7] は各タスク毎に教師データ数の割合をスケジューリングすることで各タスクの学習進度を調整している．しかし本研究の場合，1 つの入力データに対し 3 タスクの出力が得られるモデルのため，このスケジューリング戦略は適用できない．本研究では，各タスクの重み係数 λ を学習の進行に伴い動的に変更して，各タスクのスケジューリングを行う．

$$\lambda(p) = \lambda_{const} \frac{1}{1 + \exp((p_{th} - p)/\alpha)} \quad (11)$$

スケジューリング関数 $\lambda(p)$ は，sigmoid 関数を用いて (11) 式の通り決定した．なお p は学習エポック数を連続量で表したものであり，エポック数 n_{epoch} 及び学習イテレーション数 i ，全イテレーション数 i_{all} を用いて $p(n_{epoch}, i) = n_{epoch} + i/i_{all}$ で求める．図 2 に示されるように，パラメータ p_{th} をタスク毎に変更することで，各タスクの学習スケジューリングを行う．λ_{const} は各タスクの損失の大小を吸収するためのパラメータであり，またパラメータ αは 0.5 と決定した．

表 3　ハイパーパラメータの一覧

語彙数	12,000	Optimizer	Adam
単語埋込層次元	300	学習率	0.001
隠れ層ユニット数	256	バッチサイズ	16
最大入力長	300	Coverage 損失重み	0.1
最大出力長	32	正規化 (Dropout)	rate=0.5
最大職種フレーズ長	8	損失関数 L2 正則化	0.0001

表 4　マルチタスク学習による見出し文生成の評価

	Rouge-L		
	P	R	F 値
リード文 (本文先頭 1 行)	0.130	0.253	0.135
見出し文のみ (ベースライン)	0.195	0.265	0.196
見出し文 + 文書分類	0.204	0.286	0.207
見出し文 + 職種フレーズ生成	0.206	0.288	0.209
見出し + 職種フレーズ + 分類	**0.208**	**0.292**	**0.212**

表 5　職種フレーズ生成の精度評価

	Rouge-L		
	P	R	F 値
職種フレーズ生成 (ベースライン)	0.223	0.288	0.219
職種フレーズ生成 + 文書分類	0.243	0.309	0.238
職種フレーズ + 見出し文生成	0.242	0.304	0.235
職種フレーズ + 見出し + 分類	**0.246**	**0.316**	**0.241**

表 6　職種別文書分類の精度評価

	accuracy
文書分類のみ (ベースライン)	0.615
文書分類 + 見出し文生成	**0.627**
文書分類 + 職種フレーズ生成	0.621
分類 + 職種フレーズ + 見出し文	0.624

4　実験

4.1　実験設定

本研究では，要約対象の記事本文および見出しに固有名詞が多く含まれるため，3.2 章で述べたコピー機構を適用した注意機構付き生成型要約を使用する．1 件の求人記事を入力として与えた時，文書分類層で職種カテゴリの分類を出力し，2 つのデコーダ層でそれぞれ職種フレーズと求人記事見出しを出力するようマルチタスク学習を行う．なお，職種フレーズが欠損しているデータはフレーズ生成デコーダを学習せず，残りの 2 タスクのみ学習している．単語埋込層には，FastText [8] で事前学習されたモデルを利用する．また，単語分割には mecab-ipadic-NEologd 辞書を適用する．

提案手法を検証するにあたって，生成型要約の実装におけるハイパーパラメータは，検証用データにおけるパラメータ探索の結果より 表 3 の通り決定した．また，学習スケジューリングのパラメータ p_{th} として，文書分類，職種フレーズ生成及び見出し文生成の損失関数パラメータをそれぞれ $p_{th}^{cla} = 0.0$, $p_{th}^{phrase} = 1.0$, $p_{th}^{title} = 2.0$ と設定した．損失重み係数パラメータは $\lambda_{const}^{cla} = 5.0$, $\lambda_{const}^{phrase} = \lambda_{const}^{title} = 1.0$ と決定した．

4.2　実験結果

マルチタスク学習は教師データの少ない本タスクの精度を向上させるか？　マルチタスク学習の効果を検証するために，マルチタスク学習の有無による比較実験を行った．生成された見出し文・職種フレーズは評価指標 Rouge-L で評価する．比較実験は 10 分割交差検証で実施した．表 4 に比較実験の結果を示す．見出し文生成だけではなく職種フレーズ生成・文書分類とのマルチタスク学習を適用することで Rouge-L の F 値が 1.6pt 向上し，より適切な見出し文が生成されたと言える．また，職種フレーズ生成・文書分類の精度評価を表 5 及び

表 7　学習スケジューリングによる見出し文生成の評価

	acc	Rouge-L F 値	
	分類	職種フレーズ	見出し
スケジューリングなし	0.623	0.230	0.209
$p_{th}^{phrase}=0, p_{th}^{title}=2$	0.617	0.238	0.210
$p_{th}^{phrase}=2, p_{th}^{title}=2$	0.624	**0.242**	0.210
$p_{th}^{phrase}=1, p_{th}^{title}=2$	**0.624**	0.241	**0.212**

表 8　人手評価の結果

	見出し文の適切性	3 タスク間の一貫性
マルチタスク学習無し	0.35	0.41
マルチタスク学習有り	**0.44**	**0.73**

表 6 に示す．職種フレーズ生成においてもマルチタスク学習の適用により精度が向上した．

学習スケジューリングは難易度差のあるマルチタスク学習を改善するか？　次に, 提案手法である学習スケジューリングの有効性を検証するため, スケジューリングのパラメータ p_{th} を変えて比較実験を行った．結果を表 7 に示す．$p_{th}^{phrase}=1.0$ 及び $p_{th}^{title}=2.0$ の設定の精度が最良となった．このことから, 学習スケジューリングにより職種フレーズ生成と見出し文生成タスクの学習を遅らせることで盗み聞き効果により学習が促進されたと言える．

マルチタスク学習は 3 タスク間での職種一貫性を改善するか？　まず, 人手評価により見出し文の適切性を検証した．出力からランダムに 100 記事分抽出し, 2 人の評価者に依頼し各記事毎に 2 段階で絶対評価を行う．その平均を適切性の評価値とした．表 8 の左側に適切性の人手評価結果を示す．マルチタスク学習の適用により見出し文の適切性が改善され, より正しい情報が含まれる適切な見出し文が生成されたとわかる．

同様にして, 人手評価により 3 タスク間での職種一貫性を検証した．職種一貫性の判定基準として, 見出し文及び職種フレーズが同一職種を言及し, かつ職種分類のカテゴリに見出し文の職種が含まれる時に一貫性があるとみなす．2 人の評価者に依頼し, ランダム抽出した 100 件の記事に対して職種一貫性の有無の 2 段階で絶対評価を行う．その平均を職種一貫性の評価値とした．なお, 生成された見出し文に職種が含まれない場合及び, 生成結果の職種が評価者に判別不能の場合は評価対象から除外している．表 8 の右側に職種一貫性の人手評価結果を示す．マルチタスク学習により職種一貫性の人手評価精度が向上し, 3 タスク間で一貫性のある分類・生成が行えたと言える．

4.3　考察

表 4 より, 見出し文生成には文書分類より職種フレーズ生成とのマルチタスク学習が有効との結果が得られた．見出し文生成タスクの学習には, よりタスクの類似度している文生成タスクの方が効果があることを示唆している．一方表 5 より, 職種フレーズ生成には見出し文生成より職種分類とのマルチタスク学習が有効との結果が得られた．文書分類も職種フレーズ生成も入力中の職種を表す特徴に着目するのに対し, 見出し文生成はそれ以外の特徴にも着目するする必要があるため, 職種分類とのマルチタスク学習の方が Attention Focusing の効果が高まることが推測される．

また表 9 に, 見出し文の生成例を示す．マルチタスク学習なしでは誤った職種を含む見出し文が生成された

表 9　見出し文の生成結果の例

Wantedly では、デザイナーも一緒にコードを書きます。 それは、エンジニアでは最後の UX でこだわれない場所を デザイナーであればこだわれるからです。(中略) サービス開発に取り組みたいという、デザイナーさん！ ぜひ一度オフィスに話を聞きに来てみませんか?
【正解】デザインもコードも書きたいデザイナーウォンテッド!
【マルチタスク無し】フロントエンドエンジニア募集！
【マルチタスク有り】急成長中のスタートアップで デザインを学びたいデザイナー募集!

表 10　3 タスクの分類・生成結果の例

マルチタスク学習無し	
見出し文	新規事業を支える！ Web ディレクター募集！
職種フレーズ	SNS 運用担当
職種カテゴリ	マーケティング
マルチタスク学習有り	
見出し文	WEB 広告の運用を担うマーケター募集！
職種フレーズ	web マーケター
職種カテゴリ	マーケティング

が, マルチタスク学習により正しい職種 “デザイナー” を含む見出し文が出力でき, 正しく職種を捉えている．これより記事本文中の職種にまつわる特徴を的確に捉えられるようエンコーダが学習されたと言える．また表 10 に, マルチタスク学習の有無による 3 タスク生成結果の例を示す．マルチタスク学習適用により職種に関する一貫性が保てるようになったと言える．

5　おわりに

本研究では, 見出し文生成・職種フレーズ生成・職種ごとの文書分類の 3 タスクにマルチタスク学習を適用し, 一貫性のある職種分類・職種フレーズ・見出し文生成を行う手法を提案した．また, 学習スケジューリングによるマルチタスク学習の性能向上を試みた．求人記事データセットによる評価実験の結果性能向上が確認され, 提案手法の有効性が示された．

今後の課題としては, (1) 学習の進展に合わせて適応的に損失関数の重み $\lambda(p)$ を算出する手法 (2) エンコーダ層だけではなくデコーダ層の情報も共有し, マルチタスク学習の効果を増大する手法を検討したい．

参考文献

[1] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

[2] Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. Extractive summarization using multi-task learning with document classification. In *EMNLP*, 2017.

[3] Mohit Bansal, Ramakanth Pasunuru, and Han Guo. Soft layer-specific multi-task summarization with entailment and question generation. In *ACL*, 2018.

[4] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

[5] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv:1706.05098*, 2017.

[6] Abigail See, Peter J. Liu, and Christopher D. Manning. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*, 2017.

[7] Eliyahu Kiperwasser and Miguel Ballesteros. Scheduled multi-task learning: From syntax to translation. *TACL*, 2018.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL*, 2017.